Universidad Autónoma del Estado de Hidalgo

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

“ASP (ACTIVE SERVER PAGES), CASO DE ESTUDIO: TRALCOM, S.A. DE C.V.”

M O N O G R A F Í A:

QUE PARA OBTENER EL TÍTULO DE LICENCIADO EN SISTEMAS COMPUTACIONALES

P R E S E N T A:

EDGAR ANTONIO GUERRERO GONZÁLEZ

ASESOR: LIC. EN COMP. LUIS ISLAS HERNANDEZ

PACHUCA DE SOTO, HGO. 2006

INDICE

ÍNDICE DE FIGURAS

Capitulo 6. Procesado de formularios.

Capitulo 7. Objetos Integrados de ASP.

Capitulo 9. Fuentes de Datos ODBC.

Capitulo 10. Bases de Datos.

Capitulo 11. Caso de Estudio: Tralcom, S.A. de C.V.

INTRODUCCIÓN

Las necesidades de las empresas, están siendo cambiadas por los continuos avances tecnológicos y exigencias actuales. En este orden de necesidades, los lenguajes de programación se han ido revolucionando, por ejemplo: la sustitución del viejo esquema de la programación estructurada, la cual es considerada como carente de funcionalidad y eficacia, que no provee todas las opciones y facilidades que el mercado actual exige, por una programación modularizada y dinámica, que soporte la carga de miles de usuarios concurrentes y la respuesta de los sistemas de manera inmediata y eficaz. Estos cambios posibilitan un diseño y maximización de recursos y herramientas para un desarrollo de aplicaciones más activo y poderoso, facilitan el desarrollo del aprendizaje y creatividad de los programadores en un tiempo mucho menor, permiten una interacción más funcional con sus clientes.

El lenguaje de programación ASP, en todos los ámbitos actuales (empresariales, educativos, etc.) a tomado una enorme relevancia por tener una gran facilidad y mucho poder para su utilización, así como permitir la creación de aplicaciones mas potentes y funcionales para las organizaciones.

Aunado a lo anterior, las organizaciones han puesto gran interés por ofrecer a todos sus clientes sistemas con tecnología informática actual, que les ofrezca una mayor satisfacción y facilidad en el uso de dichos sistemas y así estar a la vanguardia en el mercado empresarial.

El presente trabajo se desarrolla en once capítulos, del primero al tercero se abordan los conceptos básicos del lenguaje, se muestran las configuraciones básicas para el uso de esta tecnología y se tratan los operadores básicos que existen en todos los lenguajes y su utilización en ASP.

Del cuarto al sexto capitulo, se abordan las funciones y estructuras de control utilizadas en el lenguaje, la forma en que mezclamos el lenguaje ASP con HTML y la interacción con el usuario.

Del séptimo al octavo capitulo, se muestran los objetos y variables, así como las configuraciones de inicio de nuestro sistema.

Del noveno al décimo capitulo, se abordan las configuraciones de acceso a base de datos, así como la creación de la misma y sus conceptos básicos de utilización y configuración de seguridad.

En el onceavo y ultimo capitulo, se muestra el caso de estudio de este trabajo, en donde se aplica lo aprendido en los capítulos anteriores, empleándolo en un caso practico y real de utilización de esta tecnología.

JUSTIFICACIÓN

La programación orientada a la tecnología Web, en la actualidad no es nada nuevo, existen una gran variedad de sitios realizados en esta tecnología, sin embargo, debido al desconocimiento del lenguaje ASP, una gran mayoría están construidos en lenguajes que no son muy potentes, como por ejemplo en HTML, y realizados principalmente por personas que no cuentan con grandes recursos y conocimientos para el desarrollo en otras tecnologías mas nuevas.

Otros sitios Web están construidos basados en otros lenguajes y tecnologías, como por ejemplo el lenguaje php que corre sobre el sistema operativo unix, sin embargo, ASP al ser un lenguaje que corre sobre el sistema Windows, es mas fácil de utilizar y cuanta además con muchísimo mas soporte técnico que otros.

Estas son las principales razones de la realización de este trabajo, mostrar la facilidad y potencia del desarrollo con el lenguaje ASP y así aprovechar en mayor grado las ventajas que esta tecnología proporciona a los desarrolladores y principalmente a los que quieren iniciarse en este lenguaje de programación.

OBJETIVO GENERAL

- El presente trabajo tiene el propósito de brindar una guía completa y con un lenguaje lo mas sencillo posible para el usuario inexperto para aprender a utilizar el lenguaje de programación ASP y realizar el desarrollo fácil de sitios Web.

OBJETIVOS PARTICULARES

- Se mostrara la potencialidad y facilidades para el desarrollo de aplicaciones Web en el lenguaje ASP, así como las ventajas que este proporciona en los sistemas realizados con esta tecnología.

- Se provocara el interés en el desarrollo de aplicaciones en este lenguaje, para una mayor modularidad y funcionalidad en los sistemas Web de las organizaciones y/o personas.

DEDICATORIA

A mi Padre

Gracias Papa por todos los consejos, tu apoyo, tu esfuerzo y tu arduo trabajo para darme lo que necesitaba, y por la confianza que tuviste en mi para poder conseguir el logro que hoy se ve reflejado en este titulo, el cual es por ti y para ti. Gracias por los regaños que me diste y que hicieron que enderezara mi camino, por la educación que me diste y que me guiara por el resto de mi vida. Simplemente gracias por que siempre estas ahí y por darme el orgullo de ser mi Padre.

A mi Madre

Gracias Mama por tus grandes y duros esfuerzos por darme todo por obtener este titulo, que hoy te digo con orgullo: es tuyo. Gracias por siempre estar al pendiente de mí y apoyarme en todos mis éxitos y tropiezos. Simplemente gracias por que siempre estas ahí y por darme el orgullo de ser mi Madre.

A mis Hermanas

Gracias hermanas por apoyarme siempre, por los consejos que me han dado y que me han ayudado en momentos difíciles y de incertidumbre. Gracias por las alegrías que hemos pasado juntos y también por las tristezas, las cuales han hecho que nuestra relación se fortalezca, las quiero mucho.

A mis Profesores.

Gracias a todos los profesores con los cuales tuve el privilegio de tomar clases, por su ayuda y por las enseñanzas y educación que me dieron para mi formación profesional. Gracias a todos por su ayuda y consejos y por los momentos que compartimos juntos.

A mi Asesor.

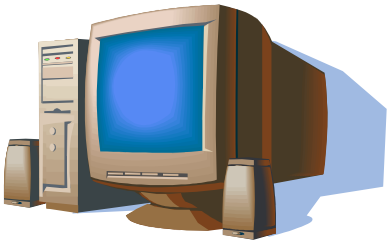
Gracias a mi asesor por su enorme ayuda para la realización de este trabajo, gracias Lic. Luis Islas por su apoyo y por todo, sin su ayuda este trabajo no fuera posible.

A mi hermano el Tijuas.

Gracias carnal por tu ayuda, este trabajo también te lo debo a ti, sin tu ayuda otra historia seria. Los amigos existen y yo lo he comprobado contigo, aunque de repente te atrofias, pero eres como mi hermano, y hemos compartido momentos excelentes de nuestra vida estudiantil y extra-escolar. Solamente te puedo decir gracias por todo.

CAPITULO 1

INTRODUCCIÓN A LA PROGRAMACIÓN EN ASP.



En este capitulo se abordan todos los aspectos básicos y necesarios para comenzar a programar en ASP.

El objetivo de este capitulo es mostrar lo que se necesita saber antes de aventurarse al aprendizaje del lenguaje ASP, por lo que al terminar de estudiar el presente capítulo podremos ser capaces de desarrollar una página sencilla en ASP.

1.1. Introducción.

Quizás hemos escuchado con frecuencia nombrar el termino *Paginas ASP* y aunque sabemos simplemente que es un lenguaje de programación, no sabemos con certeza qué es, cuales son los conceptos básicos, sintaxis o cómo y para que utilizar este lenguaje de programación.

ASP es un lenguaje de programación que se ejecuta del lado del servidor, este tipo de programación existe desde los comienzos de Internet, aunque en ese entonces se utilizaban tecnologías más rudimentarias. En concreto se utilizaba la tecnología CGI (Common Gateway Interface, pasarela de interface común), que básicamente son programas independientes escritos en algún lenguaje de programación(C, Pascal, Fortran, etc.) que ejecutan mandatos para generar una salida HTML. Esta tecnología no ha desaparecido, sigue existiendo, pero cada vez se encuentra mas en desuso por los inconvenientes que presenta.

Con la aparición de lenguajes script, apareció también el concepto de *script de servidor,* que son trozos de código (que se mezclan con el lenguaje HTML) que se ejecutan en el servidor. Con lo anterior ya no era necesario construir un programa en un lenguaje *duro* de programación, solo se tendrían que escribir unas cuantas líneas de código dentro del archivo HTML para que el servidor lo ejecutara.

Este lenguaje es parte de una tecnología impulsada por Microsoft desde hace ya varios años y que en la actualidad es uno de los lenguajes de programación Web mas utilizados. La primera versión surgió en diciembre de 1996 y fue la versión 1.0, la cual se ejecutaba sobre IIS 3.0 (el servidor Web que se vera en el siguiente capitulo). Después surgió la versión 2.0 que corría sobre IIS 4.0 y la versión mas actual es la 3.0, la cual corre sobre IIS 6.0. Por ultimo existe una nueva versión llamada ASP .NET que es parte de una filosofía nueva de desarrollo de Microsoft.

ASP tiene muchas ventajas: la facilidad para conectarse a bases de datos, lo que hace que sea muy fácil mostrar grandes catálogos por Internet (por ejemplo), la ventaja de poder tener un Web Site dinámico, es sencillo de aprender para el que tiene experiencia con cualquier otro lenguaje de programación.

1.2. Conceptos Básicos.

Las siglas ASP corresponden a las palabras ***Active Server Pages*** (Páginas Activas del Servidor), es un entorno que sirve para crear y ejecutar aplicaciones dinámicas e interactivas en la Web, apoyándose de scripts ejecutados en el servidor.
Una definición sencilla de una pagina ASP seria: “es una mezcla entre una pagina HTML y un programa (script) que da como resultado una pagina en HTML que es enviada al explorador del cliente”.

Con páginas simples de HTML, el cliente pide una página de un servidor (*www.elquesea.com*), el servidor la envía y la página es mostrada tal cual en el explorador.

Las páginas ASP en cambio, ejecutan los stripts por el servidor antes de ser enviados. El servidor los procesa mediante la dll llamada asp.dll, que es la que interpreta los mandatos ASP, y le regresa al cliente una pagina HTML sin código de servidor.

Como Funciona.

Como funciona una página ASP paso a paso:

- Un usuario por medio del explorador solicita una pagina ASP.
- La solicitud llega al servidor en donde se encuentra alojada la página que se pidió.
- El servidor procesa la página y devuelve código HTML.
- El usuario visualiza la página en su navegador.

Figura 1.1 Ejemplo del flujo de una pagina ASP.

La Figura 1.1 ejemplifica el flujo del esquema descrito anteriormente, podemos notar que para el usuario no existe diferencia entre ASP y HTML porque a su explorador siempre llega código HTML puro, el que requiere realizar un trabajo extra es el servidor, ya que tiene que procesar el código ASP y transformarlo en HTML para enviarlo al cliente.

Ventajas.

Las principales ventajas del lenguaje son:

- *Mayor Seguridad:* Al ejecutarse en el servidor, el código fuente nunca es enviado al explorador, con lo que el usuario no puede obtener dicho código fuente de nuestras paginas.
- *Mayor Funcionalidad:* Al ejecutarse en el servidor, se pueden realizar cosas que no se podrían en el cliente, como guardar datos en una base de datos, compartir datos entre distintos usuarios (por ejemplo para realizar un contador de visitas), entre otras.
- *Mayor Compatibilidad con los Exploradores:* Al ejecutarse en el servidor, se pueden generar páginas que contengan exclusivamente HTML, con lo cual no forzamos a que el explorador sea alguno que soporte el código.

- *Lenguaje más Fácil:* ASP se suele programar en VBScript, el cual es un lenguaje casi idéntico a Visual Basic, lenguaje muy popular por su facilidad y rapidez de aprendizaje.
- *Multi-lenguaje:* ASP es realmente una plataforma de soporte para diferentes lenguajes script en el servidor, lo que provoca que aparte de VBScript también podamos utilizar otros lenguajes como JavaScript, PerlScript, Rexx, Pitón, entre otros. Esto permite a un programador utilizar el lenguaje script que sepa manejar y utilizarlo en lugar de VBScript, con lo cual no esta obligado a programar en un solo lenguaje.

1.3. Sintaxis.

Como sucede con otros lenguajes de programación, ASP sigue una sintaxis en la declaración del código, dado que ASP se encuentra embebido dentro del código HTML, es necesario indicarle al servidor que partes están escritas en un lenguaje y cuales en otro. Al igual que en HTML, el código ASP tiene un tag de inicio y uno de fin, se utilizan los siguientes delimitadores: **<%** y **%>**, utilizado de la siguiente forma:

<% Líneas de Código… %>

ASP utiliza principalmente el lenguaje VisualBasicScript que no es más que una derivación del Visual Basic. Sin embargo, es posible programar paginas ASP en JavaScript, en este trabajo nos centraremos en utilizar VisualBasicScript. Lo único que tenemos que hacer es especificar al principio de la pagina (antes de cualquier otra expresión) que tipo de lenguaje queremos utilizar. Lo anterior es una sentencia de declaración opcional del lenguaje que se quiere utilizar, lo cual se hace de la siguiente manera:

<% @ LANGUAGE="VBSCRIPT" %> Si utilizamos VisualBasicScript en servidor para programar en ASP

<% @ LANGUAGE="JSCRIPT" %> Si utilizamos JavaScript en servidor para programar en ASP

La sintaxis del VBScript es muy parecida a la de Visual Basic. Las características más notables son:

- No hay distinción entre mayúsculas y minúsculas.
- Las instrucciones terminan con un retorno de carro.
- No es necesario declarar las variables antes de usarlas (este tema lo trataremos en el punto 1.5).
- Las cadenas de texto se delimitan entre comillas dobles: “”
- Los comentarios comienzan con una comilla simple: ‘ y terminan al final de una línea (este tema lo trataremos en el punto1.4).

Existe una versión de VisualBasicScript en el lado del cliente y otra en el lado del servidor, pero las sentencias y la sintaxis son prácticamente las mismas.
Dentro de los delimitadores de ASP se puede ejecutar cualquier instrucción, expresión,

procedimiento u operador valido del lenguaje. El siguiente es un ejemplo de lo descrito anteriormente:

```
<%@LANGUAGE="VBSCRIPT"%>
<HTML>
<BODY>
Hola, bienvenido a mi página, estamos a: <%=Now( )%>
</BODY>
</HTML>
```

La función *Now()* de VBScript devuelve la fecha y hora actual del servidor. Cuando el servidor procese la página nos devolverá el siguiente resultado al explorador:

Hola, bienvenido a mi página, estamos a: 4/1/2000 14:25:55 PM

Como vemos en el ejemplo, el cliente no recibe el código ASP, si no que recibe el resultado de la ejecución de dicho código.

1.4. Comentarios.

Como ya vimos, para introducir bloques de sentencias hay que escribir los símbolos reservados:

```
<% sentencias %>
```

Pudiendo ser sentencias cualquier expresión del lenguaje, en este punto cabe señalar que las sentencias en VBScript no se separan por punto y coma (;).
Al igual que en otros lenguajes, es posible poner comentarios en el código, dichos comentarios tienen el ámbito de una línea y se realizan poniendo una *comilla simple (')* antes de la línea que queremos comentar o utilizando la palabra reservada *rem*. De esta manera toda la línea indicada como comentario, el intérprete del servidor no lo leerá ni lo ejecutara, por lo que no será enviado al explorador del cliente. Un ejemplo sencillo de cómo hacer comentarios es el siguiente:

```
<HTML>
<TITLE>Comentarios en ASP</TITLE>
<BODY>
<%
'Este es un ejemplo de un comentario utilizando una comilla simple.
Rem Este es otro ejemplo de un comentario utilizando la palabra reservada rem.
%>
</BODY>
</HTML>
```

Si notamos, los comentarios también se indican dentro de los delimitadores <%%> para que sean interpretados por el servidor sin que se devuelvan al cliente. Los comentarios son muy útiles cuando tenemos cientos o miles de líneas de código y queremos hacer alguna modificación, estos nos facilitarían la búsqueda del trozo de código que estemos buscando, pero debemos tener cuidado con ellos, ya que si se abusa de los comentarios, estos pueden llegar a ser una problema y un verdadero estorbo en la legibilidad de nuestro programa.

1.5. Variables.

Una *variable* es una ubicación de almacenamiento con nombre dentro de la memoria de la computadora que contiene datos, como un número o una cadena de texto. A los datos contenidos en una variable se les llama *valor* de la variable. Las variables ofrecen una manera de almacenar, recuperar y manipular valores mediante nombres que ayuden a entender lo que hace la secuencia de comandos.

Declarar las variables en ASP es una cosa muy sencilla, porque VBScript al igual que JavaScript, es un lenguaje débilmente tipificado, es decir, que todas las variables son iguales así que todas la variables son de tipo *Variant,* por lo cual no hay que declarar si se tratan de números, cadenas de caracteres, boléanos, etc., con lo que pueden cambiar su tipo en base al valor que se les asigne. Esto tiene la ventaja de que podemos reciclar variables, con lo cual podemos usar la misma variable para 2 o mas cosas diferentes a lo largo de la pagina, sin importar que estas cosas sean de un tipo diferente, lo que nos ayuda para ahorrar memoria.
Para declarar una variable en VBScript se utiliza la palabra reservada *Dim* (igual que en Visual Basic). El siguiente fragmento de código es un ejemplo de declaración de variables:

```
<%@LANGUAGE="VBSCRIPT"%>
<%
'Declaramos variables con Dim.
Dim miNombre, miEdad

'Asignamos valores a las variables declaradas.
miNombre = "Edgar Guerrero"
miEdad = 26
Dim mayoriaEdad
mayoriaEdad = 18
If miEdad > mayoriaEdad Then
 Response.Write "Hola " & miNombre & ", ya eres mayor de edad"
Else
 Response.Write "Eres menor de edad"
End If
%>
```

En este ejemplo se puede ver que al declarar la variable no especificamos de qué tipo va a ser. Esto hace que la programación sea más rápida y sencilla, pero tiene el problema de que nos podemos equivocar cuando una variable sea cadena o número, el ejemplo de esto lo podemos ver en el siguiente ejemplo:

```
<%
...
'Dim miEdad
'Esta variable es de tipo entera:
miEdad = 26
'Esta variable es de tipo cadena:
miEdad = "26"
...
%>
```

En el ejemplo anterior vemos que la variable *miEdad* no se declara de que tipo es, sin embargo la primera vez que se le asigna un valor, este es de tipo entero ya que no se encuentra entre *comillas*, la segunda vez en la que se le asigna un valor, aunque es el

mismo, al estar entre *comillas*, el lenguaje lo toma como una cadena. En el segundo caso al realizar comparaciones u operaciones aritméticas con la variable, al ser de tipo cadena nos generaría errores.

El lenguaje tiene para solucionar lo anterior la función *Is*. Esta función nos dice de que tipo es el valor que tiene asignado una variable, se tienen varios tipos: *IsArray*, *IsNull*, *IsEmpty*, *IsDate*, veamos el ejemplo de lo anterior:

```
<%
Dim miNombre
miNombre ="Edgar Guerrero"
'Aplicamos la función Is para verificar si la variable es un numero o si esta vacía.
'Mostramos el resultado y en este ejemplo regresara un False, ya que la variable es de tipo 'cadena.
Response.Write(IsNumeric(miNombre))
Response.Write(IsNull(miNombre))
%>
```

Como ya vimos en los ejemplos anteriores, podemos imprimir en pantalla el valor de nuestras variables, para eso utilizamos el objeto *Response*. También podemos mezclar texto con nuestras variables al momento de mostrarlas o imprimir varias cadenas, utilizando para esto *&* como separador, el siguiente trozo de código muestra el ejemplo de esto:

```
<%
Dim miEdad, miNombre
miEdad = 26
miNombre = “Edgar Guerrero”
'Imprimimos en pantalla las variables.
Response.Write(“Me llamo “ & miNombre & “ y tengo “ & miEdad  & “ años.”)
%>
```

Ya se mostró como declarar variables, pero dicha declaración es *opcional*, con lo cual podemos utilizar todas las que necesitemos a lo largo del código, sin tener que declararlas, sin embargo es conveniente declarar todas las variables antes de utilizarlas ya que evita posibles errores y nos facilita la lectura de nuestro código.

Para forzar a que en el código se declaren las variables de una pagina, se utiliza la función *Option Explicit*, lo cual veremos en el siguiente ejemplo:

```
<%@LANGUAGE="VBSCRIPT"%>
<%Option Explicit%>
‘Declaramos las variables y les asignamos un valor.
Dim nombre, apellido, email
nombre = "Edgar Antonio"
apellido = "Guerrero"
email = "edgarg@tralcom.com.mx"
%>
<BR>
<%Response.Write("Nombre: " & nombre)%>
<BR>
<%Response.Write("Apellido: " & apellido)%>
<BR>
<%Response.Write("Email: " & email)%>
```

En este ejemplo, al poner la línea *Option Explicit*, estamos forzando a declarar las variables que utilizamos en el código. Con dicha función, si no declaramos una variable y

tratamos de utilizarla, nuestro programa nos marcara errores por no hacerlo.
Poner esta opción evita errores tan comunes como que en una parte de la página declaremos una variable como "conexión" y en otra parte como "coonexion"

Restricciones de Nombre de las Variables

Los nombres de variables siguen el estándar de denominación en VBScript. Los nombres de las variables deben cumplir los siguientes requisitos:

- Debe comenzar con un carácter alfabético.
- No puede contener caracteres reservados del lenguaje(puntos, operadores aritméticos, etc.), siendo los caracteres validos cualquier carácter alfabético o numérico, así como el carácter _
- No debe superar los 255 caracteres de longitud.
- Debe ser única en el alcance en donde se declara.

Alcance y Vida de las Variables

El alcance de una variable se determina al declararla. Cuando se declara una variable dentro de un procedimiento, solo el código que este dentro de ese procedimiento puede tener acceso o cambiar el valor de esa variable, se llama *variable de nivel local*. Si se declara una variable fuera de un procedimiento, su valor es accesible y modificable desde cualquier procedimiento de una pagina ASP, se llama *variable de nivel global.*

La vida de una variable de nivel local va desde el momento en que se declaro hasta el momento en que finaliza la ejecución del procedimiento donde fue declarada, cuando se termina de ejecutar dicho procedimiento, la variable se destruye. La vida de una variable de nivel global va desde que se declara (fuera de cualquier procedimiento) hasta el momento en que se termina de ejecutar toda la secuencia de comandos de nuestra página.

1.6. Crear Páginas ASP.

Un archivo ASP es un archivo de texto con la extensión .asp que contiene cualquier combinación de lo siguiente: texto, lenguaje HTML, lenguaje de servidor.
Una forma rápida para crear un archivo .asp, es cambiando la extensión de los archivos HTML (.html o .htm) por la extensión .asp. Si el archivo no contiene lenguaje ASP, el servidor no utiliza el intérprete de código de servidor y envía el archivo al cliente tal cual. Como programador, esta opción es de gran utilidad, ya que se les puede poner a los archivos la extensión .asp, incluso si se piensa en agregar lenguaje de servidor mas adelante.

Para publicar una pagina ASP en Web, guarde el archivo .asp en el directorio virtual del sitio, asegurándonos que la carpeta tiene permisos necesarios para ejecutarla. A continuación, escribimos en la barra de direcciones de nuestro explorador la dirección

URL del archivo para pedirlo. Cuando el archivo se cargue en el explorador, podremos observar que el servidor nos regreso una pagina HTML. Al principio esto puede ser confuso, pero hay que recordar que el servidor ejecuta todo el código ASP de servidor antes de contestar nuestra petición, por lo que el usuario siempre recibe simple y sencillo código HTML estándar.

1.7. La Primera Página en ASP.

Ya vimos los conceptos básicos de una pagina ASP, así que podemos comenzar a programar nuestra primera página en este lenguaje, hagamos un ejemplo sencillo para comprenderlo fácilmente:

```
<HTML>
<HEAD>
<TITLE>Mi Primer Página en ASP</TITLE>
</HEAD>
<BODY>
<BR><BR>
<%
  'Dim miNombre, miEdad, Email
  miNombre = "Edgar Guerrero"
  miEdad = 26
  Email = "edgarg@tralcom.com.mx"
%>
<TABLE>
<TR>
<TD>
<FONT FACE="TAHOMA" COLOR="BLACK" SIZE="2">Nombre:</FONT>
</TD>
<TD>
<INPUT TYPE="TEXT" ID="Nombre" NAME="Nombre" VALUE="<%=miNombre%>">
</TD>
</TR>
<TR>
<TD>
<FONT FACE="TAHOMA" COLOR="BLACK" SIZE="2">Edad:</FONT>
</TD>
<TD>
<INPUT TYPE="TEXT" ID="Edad" NAME="Edad" VALUE="<%=miEdad%> años">
</TD>
</TR>
<TR>
<TD>
<FONT FACE="TAHOMA" COLOR="BLACK" SIZE="2">Email:</FONT>
</TD>
<TD>
<INPUT TYPE="TEXT" ID="Email" NAME="Email" VALUE="<%=Email%>">
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

La página tiene dos partes: la primera contiene variables en código de servidor y la segunda son etiquetas y cajas de texto de HTML que muestran el contenido de las variables declaradas en nuestro código ASP.
Este es un ejemplo sencillo de una página, cuyo principal objetivo es mostrar como se mezcla el código ASP y el código HTML.

1.8. Requisitos para ejecutar Páginas ASP.

Software.

Para que un usuario pueda ejecutar una aplicación desarrollada en ASP no se necesita de software específico, pues como ya lo vimos anteriormente, las páginas se ejecutan del lado del servidor, esto significa que no importa el navegador o sistema operativo que tenga el usuario, puesto que vera una simple pagina rendereada de HTML.

Para el funcionamiento de un servidor Web de alto rendimiento que soporte ASP, el software recomendado es:

- Sistema Operativo: Windows 2003 Server o XP
- Internet Information Server 6.0(IIS)
- Para desarrollar las páginas podemos utilizar el editor de texto de nuestra preferencia, desde el simple *notepad*, hasta software como *DreamWeaver de Macromedia* o *Visual InterDev de Microsoft Visual Studio.*

Hardware.

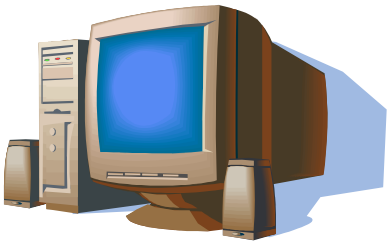
Los requisitos hardware, dependen de las exigencias específicas del servidor, del número potencial de usuarios conectados, de eventuales interconexiones con bases de datos. Es conveniente tomar en cuenta las posibles aplicaciones que pudieran reducir el rendimiento del servidor.

El hardware recomendado es:

- Intel Pentium 166MHz mínimo.
- 64 MB RAM mínimo.
- 60 MB de espacio en disco duro para la instalación de IIS.
- De 2GB a 6GB de espacio libre mínimo en disco duro para el Caching

CAPITULO 2

Internet Information Server (IIS).



En este capitulo se mostrara como utilizar el software del IIS, como instalarlo, los servicios y la seguridad que proporciona para la instalación de aplicaciones desarrolladas con ASP.

El objetivo de este capitulo es mostrar al lector lo necesario para utilizar el Internet Information Server y poder montar un sitio Web desarrollado en ASP, por lo que al terminar de estudiar el presente capítulo podremos ser capaces de Instalar y Administrar nuestra aplicación desde el IIS.

2.1. Introducción.

Para utilizar nuestra aplicación ASP podemos utilizar diferentes plataformas como Unix, Linux, etc., y diferentes servidores Web como Apache, Planet, Zeus, entre otros, pero es necesario utilizar algún producto externo para traducir el ASP antes de ser enviado al explorador del cliente. Con el IIS no es necesario ningún producto intermedio, ya que se puede utilizar directamente para nuestras aplicaciones, además de ser funcional y muy robusto, por lo que en el presente capitulo se limitara a hablar acerca de el.

Windows 95, 98, las versiones Home de XP y ME no permiten la instalación de IIS, para estos podemos probar instalar otra versión del servidor Web de Microsoft, el Personal Web Server, sin embargo no es lo suficientemente versátil para ser utilizado como servidor de un sitio de un tamaño mediano, por lo que puede ser utilizado principalmente para realizar pruebas del sitio que se vaya a desarrollar de manera "local". El PWS es recomendable principalmente para principiantes que necesitan hacer pruebas constantes.

IIS (Internet Información Server) es el servidor de páginas Web de Microsoft que proporciona comunicaciones en Internet. Este software viene en las versiones de Windows que están basadas en NT (Windows 2000 Profesional, Server, Windows 2003 Server, así como XP en las versiones Profesional y Server). Este servicio convierte a una computadora en un servidor de Internet o Intranet, por lo que se pueden publicar páginas Web de manera local y remota

En los últimos años, IIS ha evolucionado bastante, convirtiendo a Windows en una sofisticada y robusta plataforma que proporciona una infraestructura confiable, manejable y escalable como servidor Web. Además proporciona a las organizaciones la disponibilidad de sus aplicaciones disminuyendo los costos de administración del sistema.

IIS utiliza un conjunto de herramientas destinadas al control de servicios de Internet como WWW, FTP, SMTP y News. Además incluye el soporte necesario para la creación de páginas dinámicas en el servidor mediante el lenguaje ASP.

IIS ofrece grandes ventajas en los siguientes aspectos:

- *Rendimiento.* El código que procesa el *Hypertext Transfer Protocol* (HTTP) se encuentra dentro del kernel, con lo cual ofrece una enorme capacidad de manejar peticiones, y soporta una gran capacidad de aplicaciones para aprovechar los servidores multiprocesador. Esto acelera la entrega de contenido estático y la ejecución de páginas dinámicas.
- *Fiabilidad y Tolerancia a Fallos.* Tiene un buen diseño en el modelo de procesos para protegerse de manera efectiva de código errante y de esta manera proteger a los sitios Web.

2.2. Instalación del IIS.

IIS lo podemos encontrar en el CD de instalación de Windows. Para instalarlo necesitamos acceder a la opción de *Instalar componentes opcionales de Windows*, lo cual podemos hacer de la siguiente manera:

1. En el Panel de control seleccionamos la opción *Agregar o quitar programas* y en la ventana que se abre pulsamos sobre el icono de la izquierda marcado como *Agregar o quitar componentes de Windows(vease figura 2.1).*
2. A continuación nos muestra la ventana para seleccionar los componentes adicionales de Windows que hay disponibles. De la lista que se nos despliega seleccionamos la opción *Servicios de Internet Information Server (IIS)*. Por defecto se seleccionan unos cuantos componentes dentro de los que se ofrece la instalación de IIS. Nosotros podemos elegir que componentes deseamos instalar dando clic en el botón *Detalles*. Entre los principales componentes disponibles se encuentran las extensiones de FrontPage, documentación, servicios adicionales de IIS, servidor de FTP (para la transferencia de archivos por este protocolo), y un servidor de SMTP (para el envío de correos electrónicos).

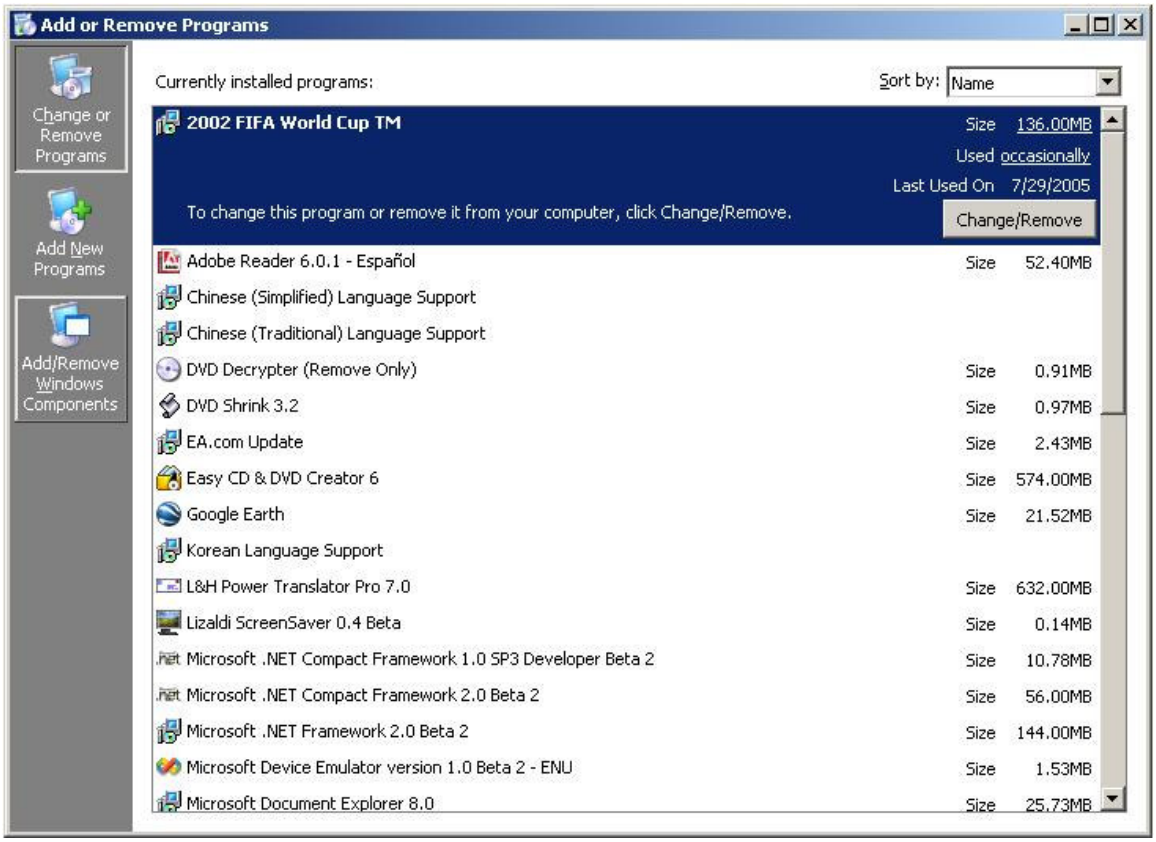


Figura 2.1 Ventana para Agregar o Quitar Programas.

Si no sabemos que componentes instalar, podemos dejar las opciones que deja el sistema por default, pues serán validas para la mayoría de los casos.

Los componentes que instala por default el IIS son los siguientes:

Componente	**Instalación por Default**
Static file support	Enabled
ASP	Enabled
Server – side incluyes	Enabled
Internet Data Connector	Enabled

WebDAV	Enabled
Index Server ISAPI	Enabled
Internet Printing ISAPI	Enabled
CGI	Enabled
Microsoft Front Page Server extensions	Enabled
Password change interface	Enabled
SMTP	Enabled
FTP	Enabled
ASP.NET	Disabled
Background Intelligence Transfer Service	Disabled

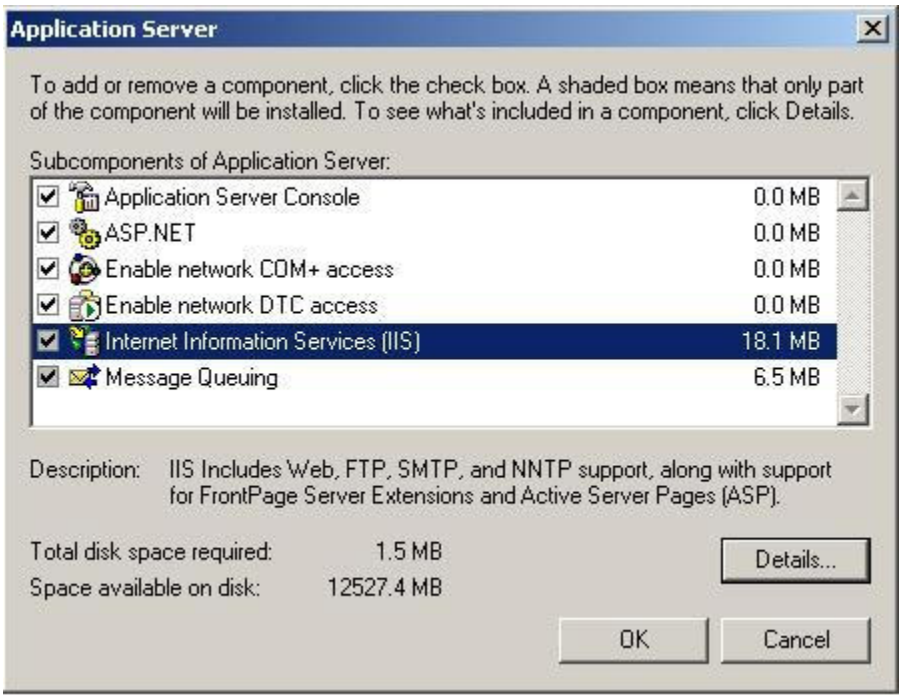


Figura 2.2 Selección de Componentes de Windows.

Una vez que hemos seleccionado los componentes que deseamos instalar(vease figura 2.2), apretamos el botón de *siguiente* para comenzar la instalación que puede durar varios minutos. Una vez terminada la instalación, tendremos instalado y listo el servidor para empezar a correr nuestras aplicaciones en ASP.

Por defecto cuando se instala el IIS se crea:

- Sitio Principal (Default Web Site).
- Sitio de Administración, donde se encuentran las ayudas y las páginas de administración del IIS en html.
- Servidor SMTP.
- El directorio Web por defecto es C:\inetpub\wwwroot

Administración de Internet Information Server.

Para administrar el servidor IIS, disponemos de un panel de control llamado *Servicios de Internet Information Server*, al cual podemos acceder de varias formas:

1 Pulsando el botón derecho del Mouse sobre el icono *Mi PC* y seleccionando la opción de *Administrar*. Esto nos abre la aplicación *Microsoft Management Console (Administración de Equipos)*. En la lista de la izquierda, en la parte de abajo aparece *Services and Application (Servicios y Aplicaciones)*, entre los que encontraremos la opción deseada: *Internet Information Server(vease figura 2.3).*

2 Desde el panel de control, si tenemos configurada la vista clásica de Windows, encontraremos un icono llamado *Administrative Tools (Herramientas Administrativas)* y haciendo doble click en ella, encontraremos el icono para administrar IIS. Si tenemos configurada la vista por categorías, la búsqueda es la siguiente: seleccionamos *Rendimiento y Mantenimiento* y dentro ya encontramos el icono *Administrative Tools*, donde encontraremos la opción para acceder al IIS.

3 Otra manera de acceder aparece en la ayuda de *Internet Information Server*. Lo que hay que hacer es ejecutar el archivo llamado *inetmgr.exe*, con lo cual aparecerá la consola de administración de IIS.

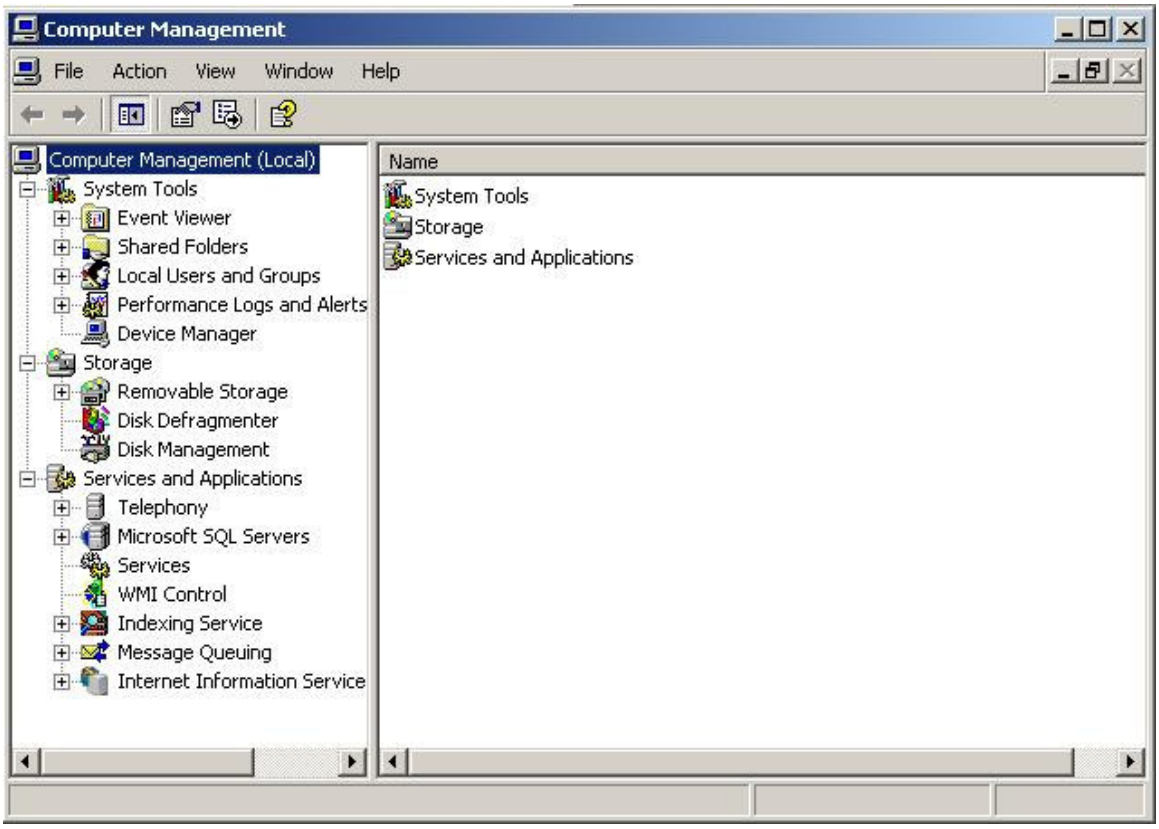


Figura 2.3 Microsoft Management Console.

Una vez que nos encontramos dentro del panel del IIS, podemos configurar nuestro servidor Web en muchos aspectos, por ejemplo, podemos definir la página por defecto, crear directorios virtuales, modificar las opciones de seguridad, etc.

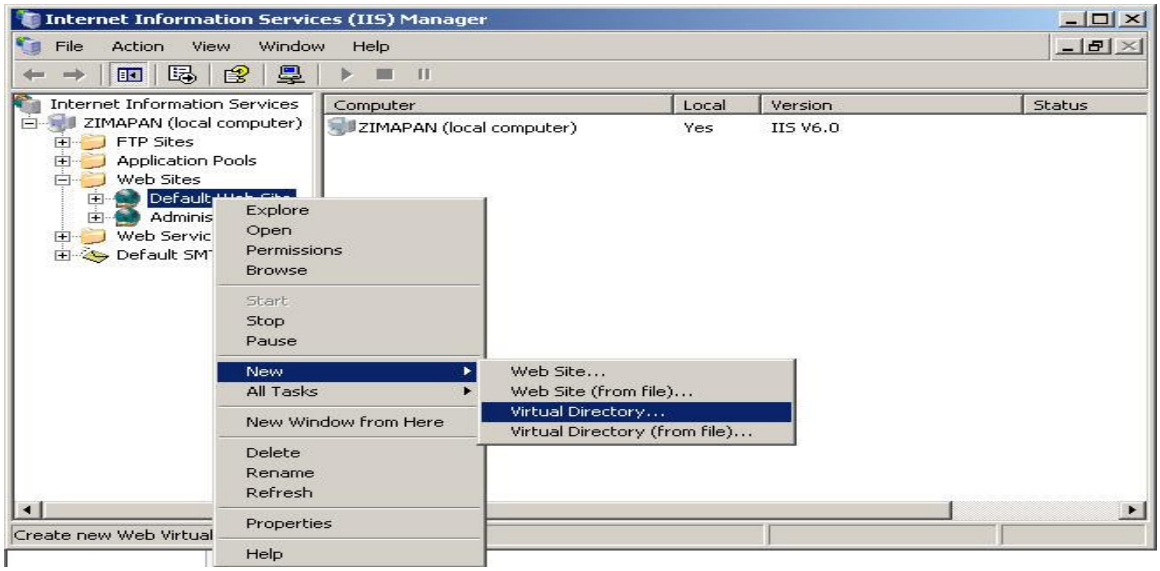


Figura 2.4 Crear un Directorio Virtual.

Directorio Virtual.

Un *directorio virtual* es un directorio del servidor que no está dentro del directorio de publicación del sitio Web, es decir, que no depende de C:\Inetpub\wwwroot (que es la ruta por default del Default Web Site o nuestro Web Site) pero que sí que se puede acceder a través del servidor Web como si estuviera dentro de dicho directorio y que sirve para esconder la localización real de nuestros archivos.

Para acceder al IIS debemos escribir una dirección como esta: http://localhost. De esta forma, se accede a la ruta C:\Inetpub\wwwroot, que es el directorio por default del IIS. El directorio virtual se accede con algo como http://localhost/directorio_virtual, pero como ya lo mencionamos, no tiene porque existir una correspondencia en disco de este directorio dentro de la carpeta del sitio, es decir, no hay necesidad de que exista la ruta C:\Inetpub\wwwroot\directorio_virtual, sino que dicha ruta podría estar en cualquier otro lugar de nuestro disco duro (por ejemplo C:\mis_paginas) o en la red local.

Para definir un directorio virtual damos click con el botón derecho del mouse sobre el sitio Web en el que queremos especificarlo y seleccionamos la opción *New > Virtual Directory...(vease figura 2.4)* Aparece un asistente que nos guiará durante el proceso.

El primer paso del asistente nos pregunta el "alias" o nombre lógico que queremos ponerle al directorio. El segundo paso nos pide la localización física de ese directorio en nuestro disco duro o en la red local. Finalmente nos solicita los permisos que deseamos asignar a ese directorio, los permisos de lectura y ejecución de código
Una vez finalizado el asistente queda creado el directorio virtual y podremos accederlo a través del alias que hayamos indicado y utiliza la pagina que se configure como predeterminada en el servidor(a continuación se hablara de este tema).

Pagina Predeterminada de Inicio.

Mencionamos algo acerca de la página por defecto en el párrafo anterior, esto significa que podemos definir la página de inicio de nuestro sitio. En IIS viene definido en los archivos default.asp, default.htm o index.htm, los cuales están estandarizados en Internet por los proveedores de hosting. Sin embargo, podemos utilizar la página de inicio que nosotros queramos, sin necesidad de utilizar alguna de las anteriores.

Lo que debemos de hacer es colocar nuestra página dentro de la carpeta del sitio Web predeterminado. Para definir la página por defecto, hacemos un click al botón derecho de nuestro mouse sobre nuestro sitio Web y seleccionamos la opción propiedades.

Aparecerá entonces la ventana de propiedades del sitio, donde seleccionaremos la pestaña marcada como *Documents* para poder predeterminar la(s) página(s) de inicio del sitio.

Podemos definir una o varias paginas de inicio, de modo que si no existe el primer archivo indicado como documento por defecto, se intentara con el segundo, tercero, etc. y así sucesivamente hasta que encuentre un archivo que mostrar o se acabe la lista. Tanto el orden como el número de archivos configurados son importantes y se pueden modificar utilizando los botones que se encuentran debajo de la lista de posibles archivos.

Si no hay ningún archivo en la lista, cuyo nombre sea alguno de los archivos por defecto, no se mostraría ninguna pagina y en su lugar recibiríamos un error 404 o el listado de ese directorio, dependiendo de cómo esté configurado IIS para este caso.

2.3. Configuración del Protocolo TCP/IP.

Se configura en el *Panel de Control* de Windows, en *Network Connections* seleccionamos la conexión de la cual disponemos (Local Area Connection) y en la ventana que se despliega configuramos los servicios, protocolos, adaptadores y enlaces.

En la pestaña *General,* seleccionamos TCP/IP de la lista que muestra, si no aparece, lo añadiremos con el botón *Install*.

Una vez escogido pulsamos *Properties* para configurarlo (vease figura 2.5).

Dirección IP:

- Dirección IP
- Mascara de subred
- Gateway

Dirección DNS:

- Dirección IP del Controlador de Dominio
- Dirección IP de un Controlador de Dominio alterno(opcional)

Todos estos datos nos los proporciona nuestro proveedor ISP.

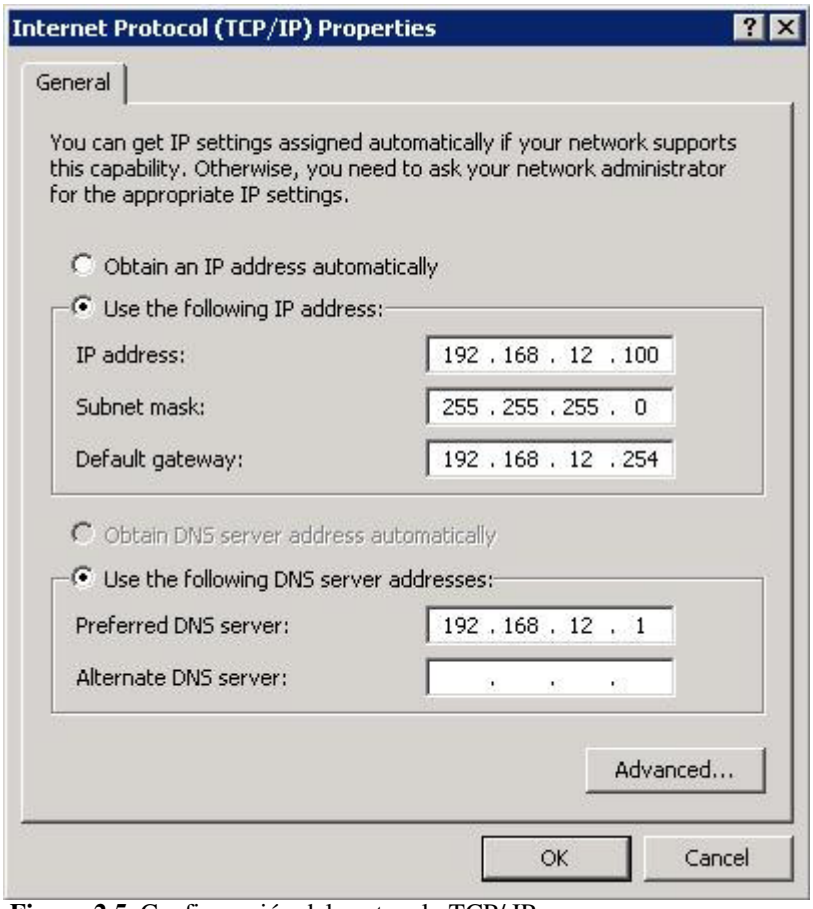


Figura 2.5. Configuración del protocolo TCP/ IP.

2.4. Seguridad.

La seguridad de un sitio Web es especialmente importante, debido a la necesidad de garantizar su utilización segura por usuarios remotos.
IIS utiliza la seguridad de Windows NT, a través de esta opción podemos dar acceso al sitio a todos los usuarios o restringiremos el acceso a todos los usuarios excepto al administrador y usuarios validados, también podemos determinar que IP tienen acceso al sitio y que IP no, dependiendo de donde procedan, y basa su seguridad en el sistema de *usuarios y contraseñas*, con lo cual logra mantener el equipo seguro. La mayor parte de las peticiones de páginas Web son realizadas por clientes anónimos, en este caso, el servidor Web se encarga de suplantar al usuario real mediante el uso de la cuenta del usuario anónimo.

Mecanismo de seguridad en una petición.

1. Comprobación de la dirección IP del cliente por IIS.
2. Comprobación de usuario y contraseña.
3. Comprobación de los permisos de acceso a archivos establecidos en el sistema NTFS.

Si cualquiera de estas comprobaciones es errónea, la petición no tendrá éxito.

Conviene revisar los derechos de los grupos que tienen los grupos *Todos* e *Invitados* a los que pertenece el usuario anónimo. Para que el usuario anónimo funcione correctamente debemos activar *Permitir Anónimos* en las propiedades del servicio Web.

Autentificación.

La Autentificación es el proceso que permite a IIS conocer las credenciales del usuario que está intentando utilizar lo servicios Web del sistema, de tal forma que solo los clientes que proporcionen un nombre de usuario y una contraseña validos tengan acceso y una vez identificado dicho usuario, entra en juego la Autorización, la cual define si dicho usuario tiene permisos o no para acceder.

En IIS existen distintas formas de autentificación:

1. *Autentificación Anónima*: Utilizando este tipo de autentificación indicamos a IIS que no solicite la identidad del usuario, y en su lugar especificamos el usuario que utilizara para atender todas las peticiones anónimas, el cual es el usuario *IUSR_Nombre de la maquina*, el cual es creado automáticamente cuando se instala IIS.
2. *Autentificación Básica*: El usuario y la clave se transmiten sin cifrar
3. *Autentificación por Respuesta de Windows NT*: El usuario y la clave se transmiten cifrados; el usuario debe de estar dado de alta en el dominio de la maquina que ejecuta IIS y tener derechos de *Acceso al equipo desde la red.*

Generalmente se permiten simultáneamente *Anónimos y mecanismos de autentificación*, en este caso en primer lugar se usa el usuario *Anónimo* y si se produce un error por falta de permisos de acceso a un recurso, el cliente recibe una ventana de dialogo solicitándole las credenciales.

Establecimiento de permisos en los directorios y ficheros de un sitio Web (aspectos Básicos).

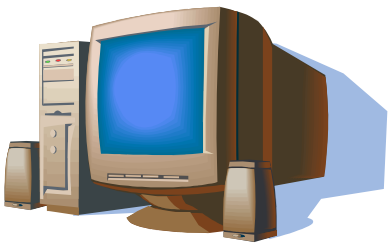
De forma genérica un sitio Web reside en:

- Un directorio particular.
- Los subdirectorios que parten del particular.
- Los directorios virtuales.

Cada uno de los elementos anteriores, en caso de existir, deberá poseer los suficientes permisos para que el sitio Web funcione correctamente, pero con las restricciones adecuadas para que el equipo este seguro y debemos asignarles permisos de Lectura y Ejecución.

CAPITULO 3

Operadores.



En este capitulo se abordaran los tipos de operadores que soporta ASP para el desarrollo de paginas.

3.1. Introducción.

Los operadores junto con las variables (las cuales vimos en el capitulo 1) y los bucles (que veremos en el siguiente capitulo), son la base de los lenguajes de programación. Las aplicaciones en ASP utilizan el control de flujo que proporcionan los operadores.

Los operadores son símbolos que sirven para designar operaciones comunes. Dichos símbolos pueden relacionar dos expresiones o utilizarse con una sola. Los operadores son necesarios para poder realizar cálculos con variables y valores constantes.

En VBScript existen todos los operadores típicos de un lenguaje de programación. Se pueden dividir en tres tipos: *aritméticos*, *lógicos* y de *comparación*.

Todos los lenguajes de programación deben establecer una jerarquía de prioridades entre sus operadores, ya que puede existir más de un operador en una misma expresión y cuando esto sucede no se pueden evaluar todos a la vez, es necesario hacerlo uno por uno. VBScript tiene las siguientes reglas para evaluar varios operadores existentes en una misma expresión:

- Cuando se utilicen paréntesis, lo primero que se evalúa será lo que se encuentre dentro de dichos paréntesis.
- Si existen operadores de varios tipos, los primeros en evaluarse serán los aritméticos, luego los de comparación y por ultimo los lógicos.
- Cuando haya dos operadores del mismo tipo de manera consecutiva, se evaluaran según las prioridades de los operadores de ese tipo.
- Si hay dos operadores de la misma prioridad, se evaluaran de izquierda a derecha.

A continuación se muestra un ejemplo donde se utilizan todos para ver como se resuelve, se supone la siguiente expresión:

*6*6+3*(3-2+5) <= 18/6 And (True Or False)*

1. Lo primero que se ejecutara, serán las expresiones entre paréntesis, ya que son las que tienen más prioridad.

2. Lo siguiente en evaluarse es la expresión *3-2+5*. como los dos operadores tienen la misma prioridad se evaluaran de izquierda a derecha, es decir, *3-2=1* y después *1+5=6*. Si se hubiera realizado la evaluación de derecha a izquierda, se hubiera tenido *2+5=7 y 3-7=-4*.

3. Después se evalúa la expresión *True Or False*, que da como resultado *True*. Hasta este momento tendremos la sig. expresión: *6*6+3*6 <= 18/6 And True* y lo siguiente a ejecutarse serán las expresiones aritméticas, ya que tiene mas prioridad que el resto de las expresiones.

4. Primero tendremos *6*6+3*6*, como la multiplicación tiene mas prioridad que la suma, se ejecutara *6*6=36* y *3*6=18*, lo cual nos daría *36+18=54*.

5. Enseguida se evalúa *18/6 = 3*, lo que nos dará como resultado la siguiente expresión: *54 <=3 And True*.

6. Lo siguiente a evaluar son las expresiones de comparación que son las siguientes en la jerarquía de prioridades. En este momento, tenemos solo una expresión de comparación que es *54 <= 3*, que nos da como resultado False. Con esto nos quedaría la expresión *False And True*.

7. Por ultimo se evalúan las expresiones lógicas, las cuales son las últimas en la jerarquía de prioridades. Así, evaluamos la expresión *False And True*, que da como resultado *False*.

3.2. Aritméticos.

Los operadores aritméticos son aquellos que sirven para realizar operaciones entre números, relacionan dos números devolviendo un resultado también numérico. Los operadores aritméticos utilizan signos numéricos para procesar operaciones matemáticas simples, entre estos signos están la suma, la resta, la multiplicación y en general cualquiera que se utilice en matemáticas.

En la siguiente lista se muestran los operadores aritméticos ordenados por nivel de prioridad:

Operador	***Nombre***	***Ejemplo***	***Descripción***
^	Exponenciación	9^2=81	Eleva un operando a una potencia.
-	Negación	-(10)	Cambia de signo un operando.
*	Multiplicación	10*9=90	Multiplica dos operándos.
/	División	97/5=19.4	Divide el primer operando entre el segundo.
\	División Entera	22\3=7	Divide dos operándos y da un resultado entero.
Mod	Modulo	11/2=1	Devuelve el resto de una división.
+	Suma	15+47=62	Suma dos operándos.
-	Resta	91-57=34	Resta dos operándos.
&	Concatenación de Cadenas	cadena1&cadena2	Concatena dos cadenas de caracteres, no es aritmético, pero se coloca en esta clasificación por su grado de prioridad, ya que se encuentra después de los operadores aritméticos y antes de los de comparación.

El nivel de precedencia indica el orden para procesar los operadores en caso de que existan más de uno en la expresión. La suma y la resta tienen el mismo nivel de precedencia, la multiplicación y la división comparten entre ellos el mismo nivel de precedencia. Cuando existen varios operadores con el mismo nivel de precedencia en una expresión, se evalúan de izquierda a derecha como ya vimos anteriormente. El orden de precedencia varía si se utilizan paréntesis en la expresión.

Veamos un ejemplo sencillo donde se muestra el funcionamiento de los operadores aritméticos:

```
<%
   Dim resultado, variable1, variable2
   Variable1 = 8
   Variable2 = 6
Response.Write("Operador Suma: Variable1+Variable2 = " & variable1 + variable2) & "<BR>"
Response.Write("Operador Resta: Variable1-Variable2 = " & variable1 - variable2) & "<BR>"
Response.Write("Operador Multiplicación: Variable1*Variable2 = " & variable1 * variable2) & "<BR>"
Response.Write("Operador Division: Variable1/Variable2 = " & variable1 / variable2) & "<BR>"
Response.Write("Operador Exponenciación: Variable1^Variable2 = " & variable1 ^ variable2) & "<BR>"
Response.Write("Operador Modulo: Variable1ModVariable2 = " & variable1 Mod variable2) & "<BR>"
Response.Write("Operador Division Entera: Variable1\Variable2 = " & variable1 \ variable2) & "<BR>"
Response.Write("Operador Negación: -(Variable1*Variable2) = " & -(variable1 * variable2)) & "<BR>"
Response.Write("Operador Concatenación: Variable1&Variable2 = " & variable1&variable2) & "<BR>"
%>
```

[Vease Anexo A1]

El programa anterior se muestra en la Figura 3.1, ya ejecutado:

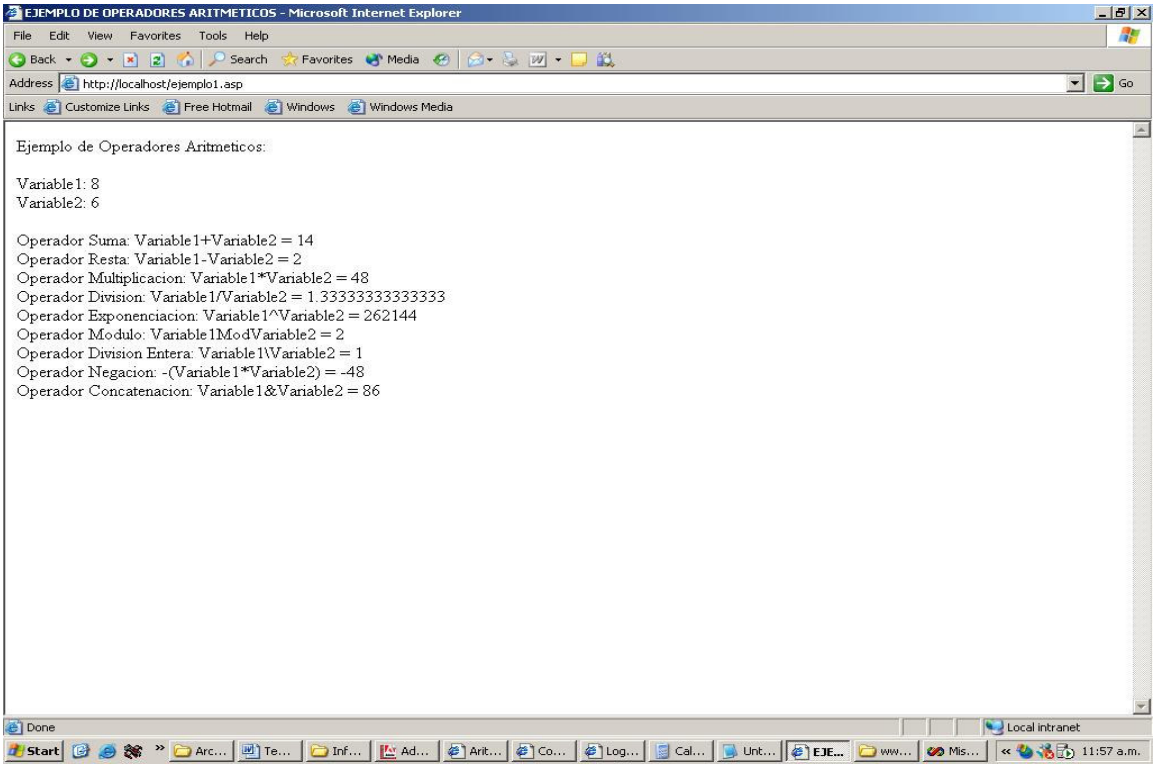


Figura 3.1 Operadores Aritméticos.

3.3. Comparación.

Los operadores de comparación sirven para comparar expresiones y verificar que cumplan alguna relación, y así obtener resultados lógicos que son de tipo booleano para poder tomar una desición.

Todos los operadores de comparación tienen el mismo nivel de prioridad y se muestran en la siguiente lista:

Operador	*Nombre*	*Ejemplo*	*Descripción*
=	Igualdad	A = B	Devuelve True si A es igual a B
<>	Desigualdad	A <> B	Devuelve True si A es diferente de B
<	Menor que	A < B	Devuelve True si A es menor que B
>	Mayor que	A > B	Devuelve True si A es mayor que B
>=	Mayor o igual que	A <= B	Devuelve True si A es menor o igual que B
<=	Menor o igual que	A >= B	Devuelve True si A es mayor o igual que B
Is	Equivalencia de objetos	IsNull(A)	Devuelve True si A es nulo

A continuación se mostrara un ejemplo sencillo del funcionamiento de los operadores de comparación:

```
<%
   Dim resultado, variable1, variable2
   Dim Igualdad, Desigualdad, Menorque, Mayorque, MenorIgual, MayorIgual, OperadorIs
   variable1 = 8
   variable2 = 6
            Igualdad = variable1 = variable2
            Desigualdad = variable1 <> variable2
            Menorque = variable1 < variable2
            Mayorque = variable1 > variable2
            MenorIgual = variable1 <= variable2
            MayorIgual = variable1 >= variable2
            OperadorIs = IsNull(variable1)
Response.Write("Operador Igualdad: Variable1 = Variable2 = " & Igualdad) & "<BR>"
 Response.Write("Operador Desigualdad: Variable1 <> de Variable2 = " & Desigualdad) & "<BR>"
 Response.Write("Operador Menor o igual que: Variable1 <= Variable2 = " & MenorIgual) & "<BR>"
 Response.Write("Operador Mayor o igual que: Variable1 >= Variable2 = " & MayorIgual) & "<BR>"
 Response.Write("Operador Mayor que: Variable1 > Variable2 = " & Mayorque) & "<BR>"
 Response.Write("Operador Menor que: Variable1 < Variable2 = " & Menorque) & "<BR>"
 Response.Write("Operador Is: IsNull(Variable1) = " & OperadorIs) & "<BR>"
%>
```

[Vease Anexo A2]

Como se puede ver en la Figura 3.2, se obtiene un valor de True o False si es que se cumple o no con la relación de cada operador.

3.4. Lógicos.

Los operadores lógicos se utilizan para realizar operaciones entre valores lógicos (de tipo booleano). Estos operadores se utilizan normalmente con dos valores boléanos, aunque puede utilizarse también con uno solo, y regresara como resultado otro valor booleano.

La siguiente lista muestra los operadores lógicos ordenados por nivel de prioridad:

Operador	*Nombre*	*Ejemplo*	*Descripción*
And	Y Lógico	(5<2) And (A = True)	Devuelve un valor verdadero cuando ambas condiciones son verdaderas.
Or	O Lógico	(B < Z) Or IsNumeric(A)	Devuelve un valor verdadero cuando al menos una de las condiciones es verdadera.

Not	Negación	Not (año = 2005)	Niega el valor de la expresión.
Xor	Or Exclusiva	(2*5 = 10) Xor (2>6)	Devuelve un valor verdadero cuando solo una de las condiciones es verdadera.
Imp	Implicación Lógica		
Eqv	Equivalencia Lógica	(20/2 = 10) Eqv (5*2 = 10)	Devuelve un valor verdadero si una de las condiciones es equivalente a la otra

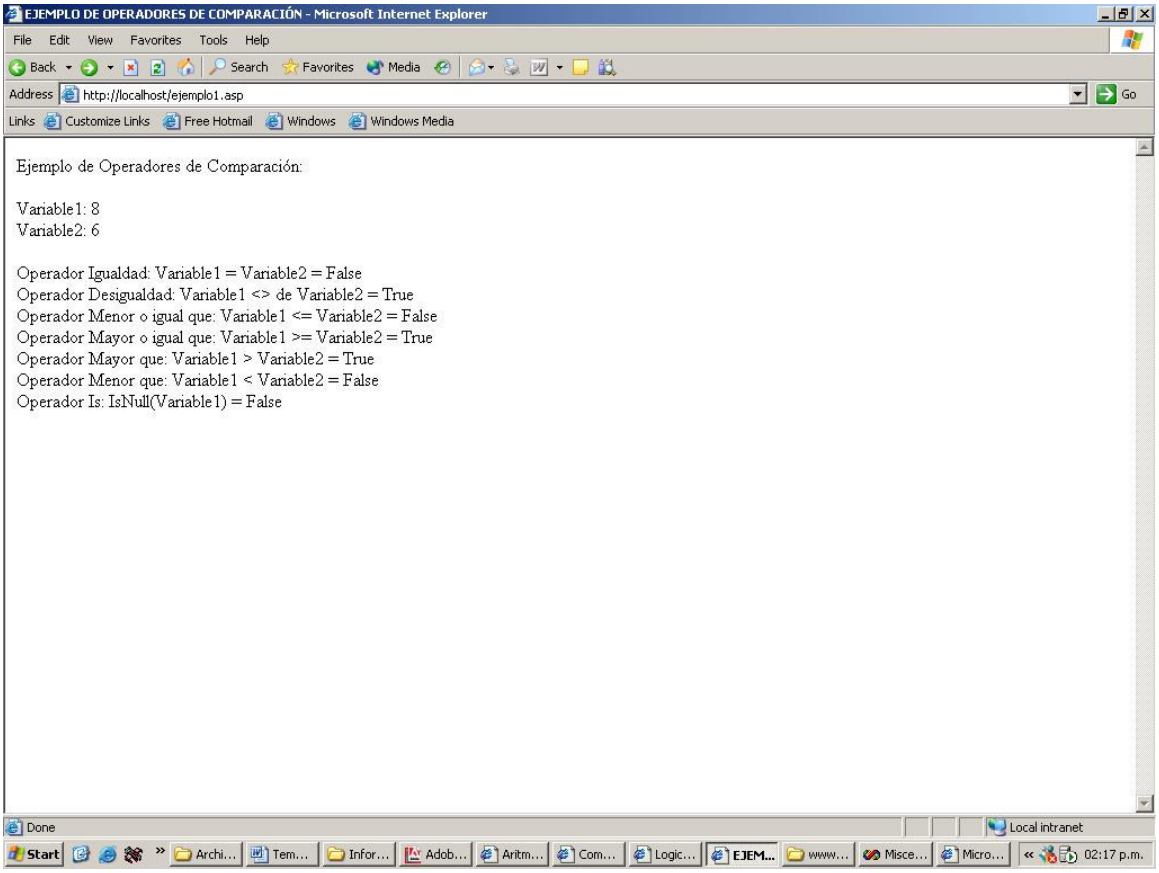


Figura 3.2 Operadores de Comparación.

Veamos otro sencillo ejemplo del uso de los operadores lógicos en el siguiente programa:

```
<%
        Dim resultado, variable1, variable2, YLogico, OLogico, OrExclusivo, Equivalencia, Negacion
        variable1 = 8
        variable2 = 6
        YLogico = variable1 = 8 And variable2 = 6
        OLogico = variable1 < 5 Or variable2 >= 6
        OrExclusivo = variable1 <= 8 Xor variable2 <> 6
        Equivalencia = variable1 * 3 = 24 Eqv variable2 * 4 = 24
        Negacion = Not (variable2 < variable1)
   Response.Write("Y Lógico: (Variable1=8) And (Variable2=6) = " & YLogico) & "<BR>"
   Response.Write("O Lógico: (Variable1<5) Or (Variable2>=6) = " & OLogico) & "<BR>"
   Response.Write("Or Exclusivo: (Variable1<=8) Xor (Variable2<>6) = " & OrExclusivo) & "<BR>"
```

```
  Response.Write("Equivalencia: (Variable1*3=24) Eqv (Variable2*4=24) = " & Equivalencia) & "<BR>"
  Response.Write("Negación: Not(Variable2 < Variable1) = " & Negacion) & "<BR>"
%>
```

[Vease Anexo A3]

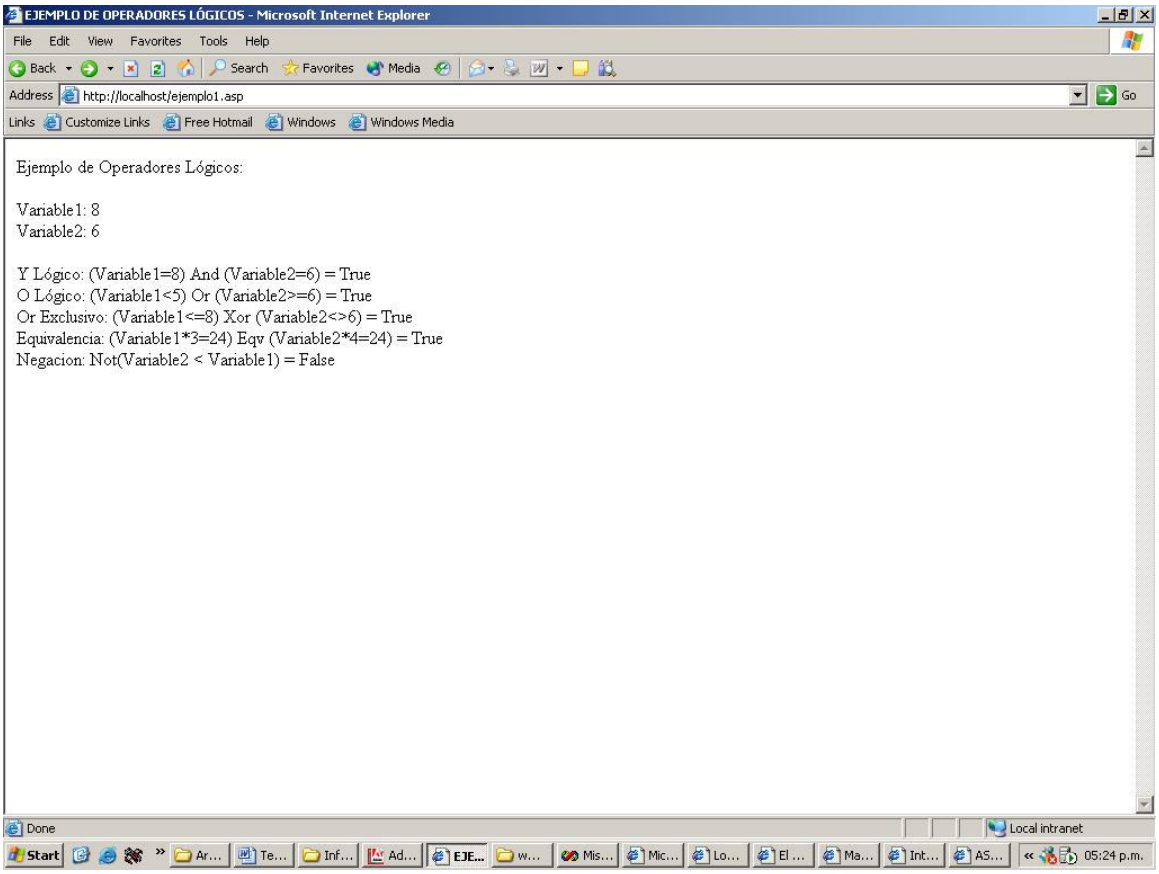
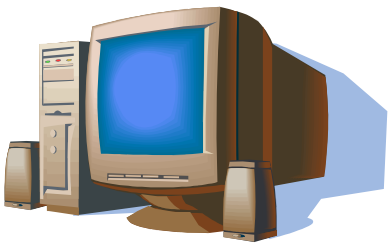


Figura 3.3 Operadores Lógicos.

En la Figura 3.3 se puede observar que al cumplirse las condiciones de las expresiones mostradas de acuerdo a la regla de cada operador, nos manda un valor de True o False. Si no se cumpliera con las condiciones, regresaría el valor contrario al mostrado en el ejemplo.

CAPITULO 4

Estructuras de Control.



En este capitulo se hablará de las estructuras de control utilizadas tanto por el lenguaje de servidor (ASP), como por los lenguajes ejecutados a nivel de cliente para potenciar la funcionalidad de las páginas ASP.

4.1. Introducción.

Las instrucciones de ASP se ejecutan de manera secuencial, es decir, cada instrucción se ejecuta una sola vez en el orden en el cual se encuentra escrita en el programa. Con esta restricción (secuencialidad escrita) no se resuelven todas las situaciones que se necesitan programar, por lo cual es necesario utilizar mecanismos que permitan variar el orden consecutivo de ejecución de las instrucciones.

En cualquier lenguaje de programación es necesario controlar el flujo de ejecución de un programa, con esto podemos hacer dinámica nuestra aplicación y ejecutar diferentes trozos de código dependiendo si cumple o no cierta condición, o agregar un bucle para ejecutar un fragmento de código repetidamente mientras se cumpla alguna condición

Los programas siguen habitualmente una regla para obtener primero datos que el usuario proporciona o de algún dispositivo de Entrada/Salida o Servidor Web, etc., se procesan dichos datos y al terminar se regresan esos datos ya procesados al exterior. Para manejar estos datos podemos utilizar las estructuras de control que el lenguaje utiliza y así poder tomar alguna desición de cómo tratar y devolver esa información.

Para realizar lo anterior utilizamos *Estructuras de Control* que son instrucciones que nos permiten realizar condiciones y en consecuencia ejecutar fragmentos de código. También las que nos permiten ejecutar repetidamente trozos de código (estas instrucciones se llaman bucles), son estructuras de control.

En VBScript existen dos tipos de Estructuras de Control: Sentencias Condicionales y Bucles, las cuales veremos a continuación:

4.2. Sentencias Condicionales.

Las sentencias condiciónales, nos permiten ejecutar instrucciones después de comprobar si una condición es verdadera o falsa. Las sentencias condicionales son la instrucción *If* y la instrucción *Select*.

Sentencia IF.

Esta instrucción nos permite elegir entre dos caminos de ejecución, dependiendo del valor de una cierta condición booleana. Si la condición booleana que se esta evaluando es cierta, se ejecutara un cierto trozo de código. Si es falsa se puede elegir entre no ejecutar nada y terminar la sentencia o ejecutar otro conjunto de código.

La sintaxis de la sentencia If es la siguiente:

```
If condicion_booleana Then
        Instrucciones a ejecutar
End If
```

En el siguiente ejemplo se muestra la utilización de la instrucción If:

```
<%
  Dim MayorEdad, MiNombre, MiEdad
  MayorEdad = 18
```

```
   MiNombre = "Edgar Antonio"
   MiEdad = 26
   If  MiEdad >= MayorEdad Then
     Response.Write("Hola " & MiNombre & "  ya eres Mayor de Edad")
   End If
%>
```

[Vease Anexo A4]

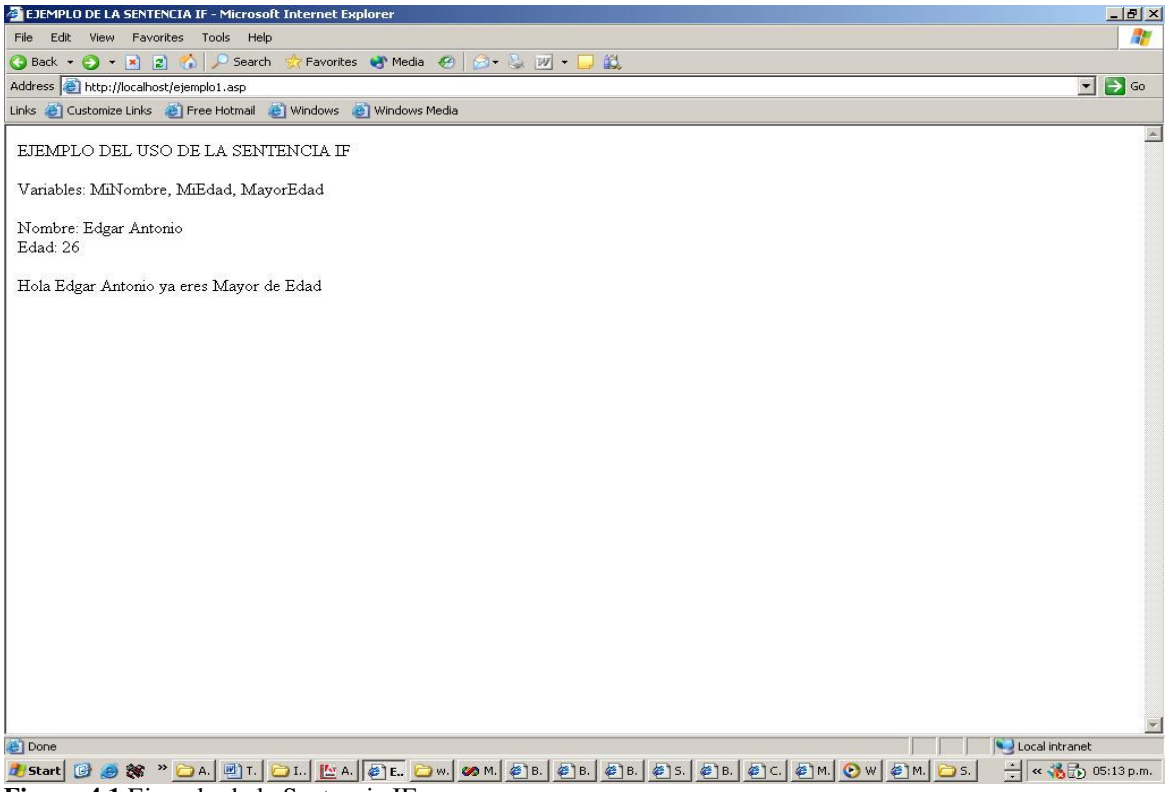


Figura 4.1 Ejemplo de la Sentencia IF.

En la Figura 4.1 se puede ver que al cumplirse la condición del If, ejecuta el código que se encuentra dentro de esta sentencia, si no se cumpliera la condición, no se ejecutaría la instrucción que se encuentra dentro del *If*.

Sentencia IF...Then...Else

La sentencia *If* puede utilizar la palabra *Else*, la cual se ejecuta cuando la instrucción *If* no cumple la condición especificada de evaluación, el lenguaje ignora todas las instrucciones que estén dentro de dicha condición, y entonces ejecuta el trozo de código que se encuentra dentro de la opción *Else*. Existirán entonces dos caminos distintos a elegir por el programa para ejecutar algún trozo de código, los cuales se elegirán dependiendo del valor que nos devuelva la condición.

La sintaxis de la sentencia If…Then…Else es la siguiente:

```
If condicion_booleana Then
        Instrucciones a ejecutar
Else
        Otras Instrucciones a ejecutar
End If
```

El siguiente Ejemplo muestra la utilización de esta instrucción:

```
<%
   Dim MiNombre, MiSexo
   MiNombre = "Edgar Antonio"
   MiSexo = "M"
 If MiSexo = "F" Then
   Response.Write("Hola " & MiNombre & " eres del sexo Femenino")
 Else
   Response.Write("Hola " & MiNombre & " eres del sexo Masculino")
 End If
%>
```

[Vease Anexo A5]

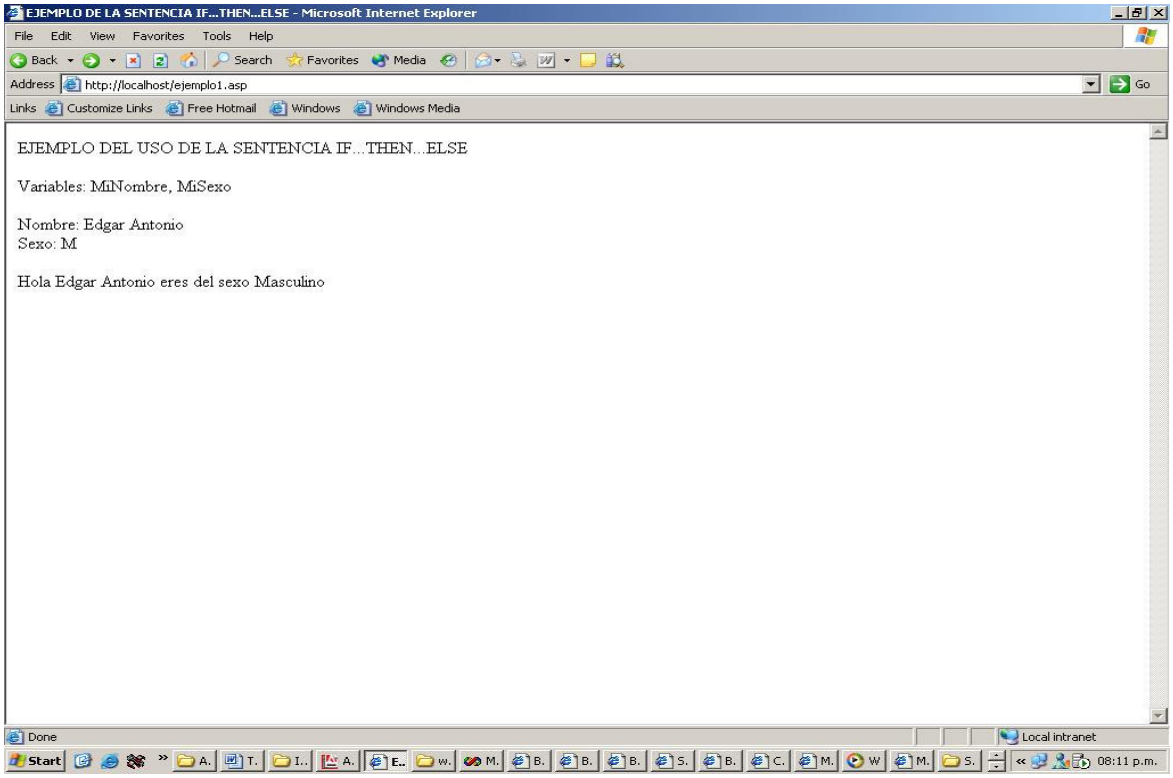


Figura 4.2 Sentencia IF…THEN…ELSE

En la figura 4.2 puede observarse que al no cumplirse la condición de la instrucción *If*, se ejecuta el código que se encuentra dentro de la opción *Else*.

Sentencia SELECT…CASE.

Cuando el número de decisiones a tomar es demasiado y además todas las opciones conciernen a la misma variable o expresión, no es conveniente utilizar la instrucción If. Para este tipo de problemas existe la instrucción *Select…Case,* la cual nos permite elegir entre un numero indefinido de opciones según el valor de una expresión.

La sintaxis de esta sentencia es muy sencilla y es la siguiente:

Select Case Expresión

```
Case valor1
  Instruccion1
Case valor2
  Instruccion2
      .
Case valorN
  Instrucción
Case Else
  InstrucciónElse
End Select
```

En esta instrucción si la expresión que se esta evaluando coincide con alguno de los valores de las opciones *Case*, se ejecuta la opción correspondiente. Si no coincide con ninguna de las opciones, se ejecuta la opción *Case Else*. La opción *Case Else* se puede omitir si se tiene seguridad de que la expresión va a caer en alguna de las opciones del Select, pero es recomendable que sea utilizada para casos de error, es decir, cuando la expresión no coincida con ninguna de las opciones posibles que tenga la sentencia.

Veamos a continuación un sencillo ejemplo del uso de esta instrucción:

```
<%
   Dim MiNombre, MiOrigen
   MiNombre = "Edgar Antonio"
   MiOrigen = "Hidalgo"
   Select Case (MiOrigen)
            Case "Chiapas"
                  Response.Write("Hola " & MiNombre & " eres Chiapaneco")
            Case "Oaxaca"
                  Response.Write("Hola " & MiNombre & " eres Oaxaqueño")
            Case "Guerrero"
                  Response.Write("Hola " & MiNombre & " eres Guerrerense")
            Case "Yucatán"
                  Response.Write("Hola " & MiNombre & " eres Yucateco")
            Case "Guanajuato"
                  Response.Write("Hola " & MiNombre & " eres del Bajío")
            Case "Querétaro"
                  Response.Write("Hola " & MiNombre & " eres Queretano")
           Case "Hidalgo"
                  Response.Write("Hola " & MiNombre & " eres orgullosamente Hidalguense")
            Case "Sonora"
                  Response.Write("Hola " & MiNombre & " eres Sonorense")
            Case "San Luís Potosí"
                  Response.Write("Hola " & MiNombre & " eres Potosino")
            Case "Sinaloa"
                  Response.Write("Hola " & MiNombre & " eres Sinaloense")
            Case "Nuevo León"
                  Response.Write("Hola " & MiNombre & " eres Norteño")
            Case Else
                  Response.Write("Hola " & MiNombre & " tienes un origen desconocido")
   End Select
%>
```

[Vease Anexo A6]

La Figura 4.3 muestra que la expresión cumple con la opción de Hidalgo, por lo que ejecuta el código que se encuentra dentro de dicha opción. Si no se cumpliera ninguna de las opciones, entraría en la opción del Else y ejecutaría el código que se encuentra en dicha opción.

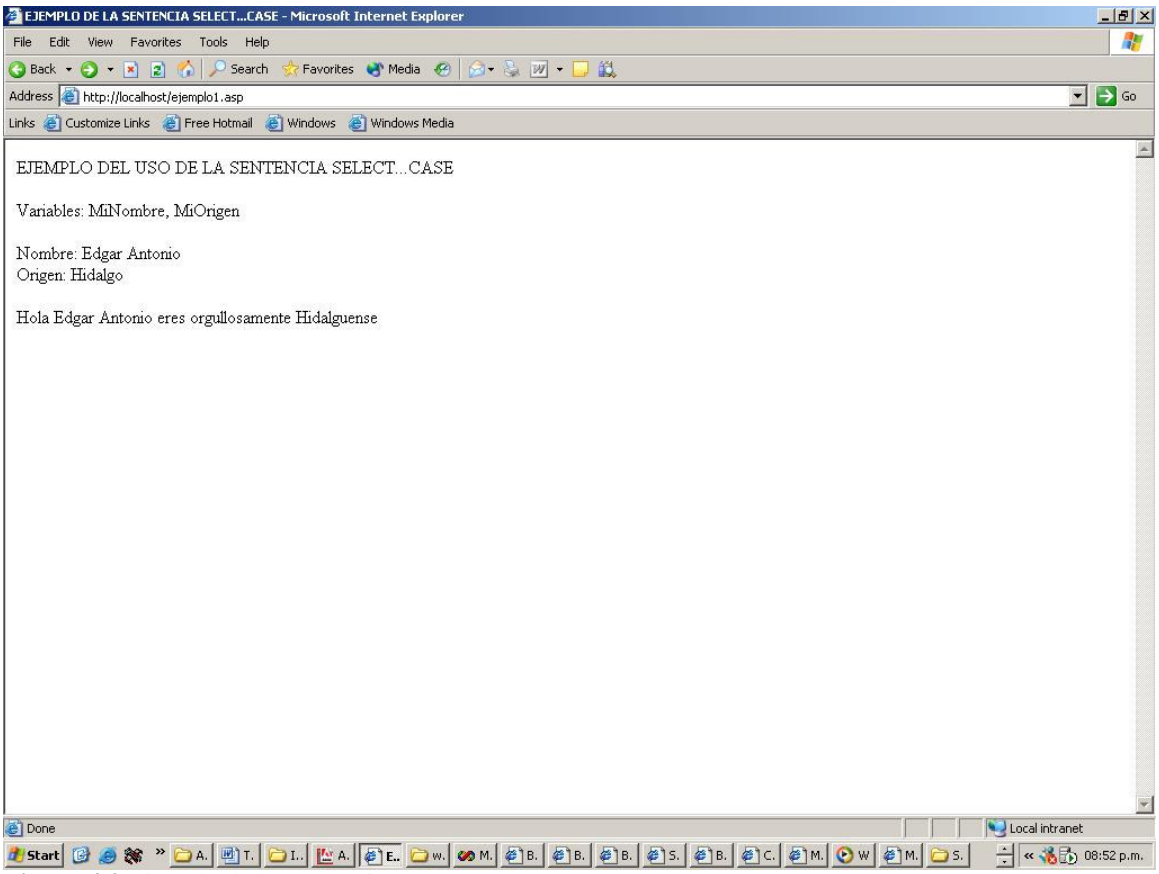


Figura 4.3 Ejemplo de la Sentencia SELECT…CASE

4.3. Bucles.

Permiten ejecutar un conjunto de instrucciones de manera repetitiva mientras se cumpla una condición. La utilización de bucles es básica en cualquier programa, ya que nos sirve para leer un archivo, obtener datos de una BD, procesar variables, entre otros.

Las instrucciones mas comunes para realizar bucles son: *For…Next, Do…Loop y While…Wend*. Estas tres instrucciones son ligeramente diferentes y la situación que se quiera resolver indicara cual es la más conveniente de utilizar. Sin embargo cada uno de estos bucles realizan la misma tarea: ejecutar un trozo de código n veces hasta que se cumpla una condición.

Bucle For…Next.

El uso de este bucle es recomendado cuando conocemos el numero de iteraciones que se deben realizar, por ejemplo si queremos realizar algún procedimiento en cada mes del año, sabremos que el numero de iteraciones necesarias será de doce.

Para utilizarlo hay que indicar el número de inicio de la cuenta, el número de terminación y el intervalo. Con esto quedara definido el número de veces que se ejecutara un trozo de

código que se encuentre entre las palabras reservadas *For* y *Next*. El valor que represente el número de iteración actual se guardara en un contador. El valor de este contador dependerá del valor de inicialización que le hayamos dado y el intervalo que cuenta. La sintaxis de esta instrucción es la siguiente:

```
For contador = valor_inicial To valor_final
        Instruccion1
        Instruccion2
              .
              .
        InstrucciónN
Next
```

Podemos utilizar este bucle de tres formas distintas. La primera es utilizando un intervalo por defecto:

```
For cont = 1To 10
  Response.Write(cont) & "<BR>"
Next
```

En este ciclo veríamos en la pantalla del explorador, escritos los números del 1 al 10. La segunda forma es utilizando la cláusula *Step*, la cual nos sirve para indicar el intervalo de cuenta que queremos utilizar, esta cláusula es optativa y puede ir o no en la construcción del ciclo. En este ejemplo si añadimos la cláusula *Step* podemos hacer que solo se vean los números pares que hay del 1 al 10 de la siguiente manera:

```
For cont = 2 To 10 Step 2
  Response.Write(cont) & "<BR>"
Next
```

La tercera forma de realizar el ciclo es haciendo que los números se impriman en orden inverso, para lo cual solo debemos poner un intervalo negativo:

```
For cont = 10 To 2 Step -2
  Response.Write(cont) & "<BR>"
Next
```

Si no se pone ningún intervalo y el numero de inicio del bucle es mayor al de finalización o el de inicio menor al de finalización y fuese negativo, no pasaría nada, simplemente el ciclo no se ejecutaría ni una sola vez.
El siguiente ejemplo muestra la utilización del bucle For en una pagina ASP:

```
<%
   Dim Meses, cont
   Meses = 12
   For cont = 1 To Meses
     Response.Write("Mes número " & cont) & "<BR>"
   Next
%>
```

[Vease Anexo A7]

La ejecución del ciclo se muestra en la Figura 4.4:

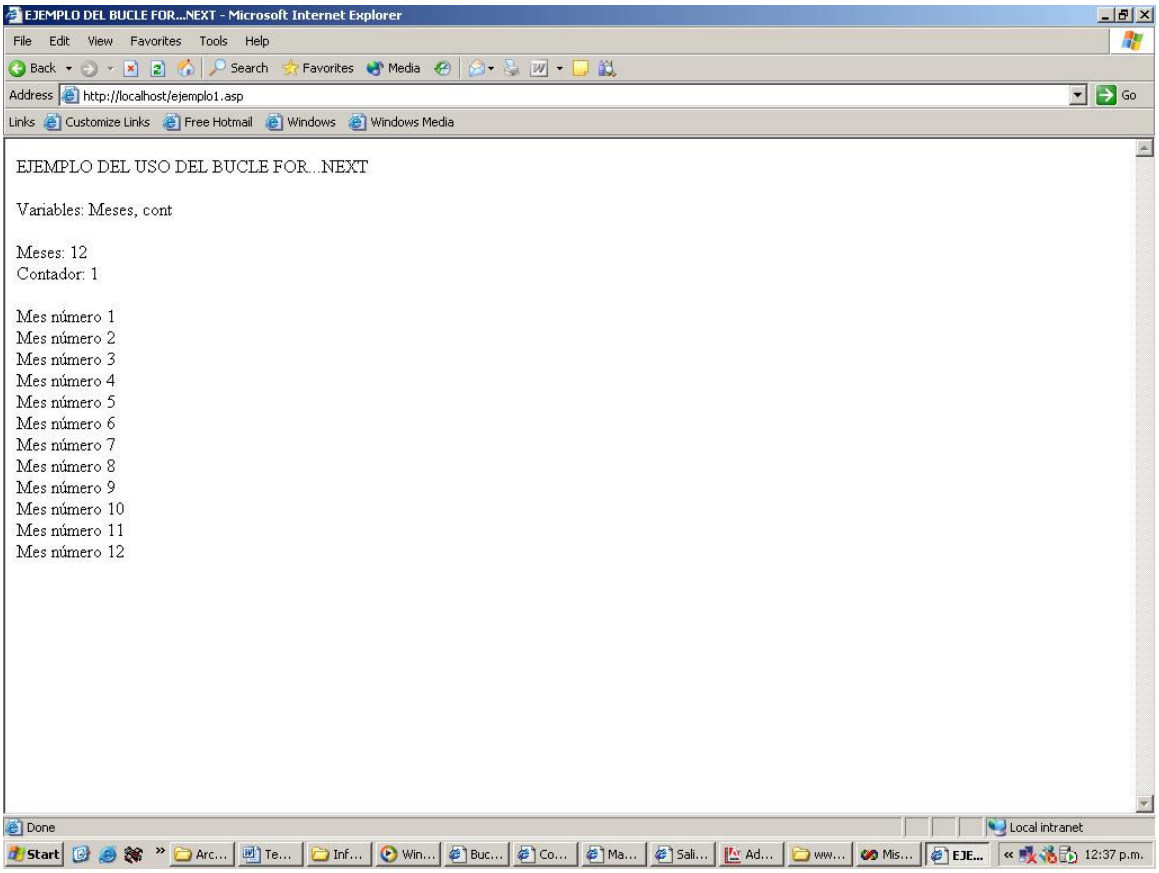


Figura 4.4 Ejecución del bucle *For…Next*

Bucle For…Each…Next.

Esta instrucción es un bucle para colecciones. Con este bucle se puede crear una iteración sobre elementos de una colección (conjunto ordenado de objetos) o de una matriz, es decir, podemos realizar la misma secuencia de instrucciones para cada elemento de la colección o matriz. Esta instrucción va a recorrer la colección con un contador, de tal forma que por cada elemento de la misma se realizaran las instrucciones que se deseen.

La sintaxis de este bucle es parecida al del anterior, solo que no hay que indicar el comienzo ni el final, ni tampoco el intervalo de cuenta, solo la colección o matriz que se va a utilizar:

```
For Each contador In colección
   Instruccion1
   Instruccion2
         .
         .
Next
```

Veamos un ejemplo del uso de esta instrucción:

```
<%
   Dim matriz(3), contador
   matriz(0) = "Edgar"
   matriz(1) = "Gisela"
```

```
   matriz(2) = "Luz Maria"
   matriz(3) = "Lizbeth"
   For Each contador In matriz
    Response.Write("Nombre: " & contador) & "<BR>"
   Next
%>
```

[Vease Anexo A8]

La Figura 4.5 muestra el ejemplo anterior ya ejecutándose en el explorador:

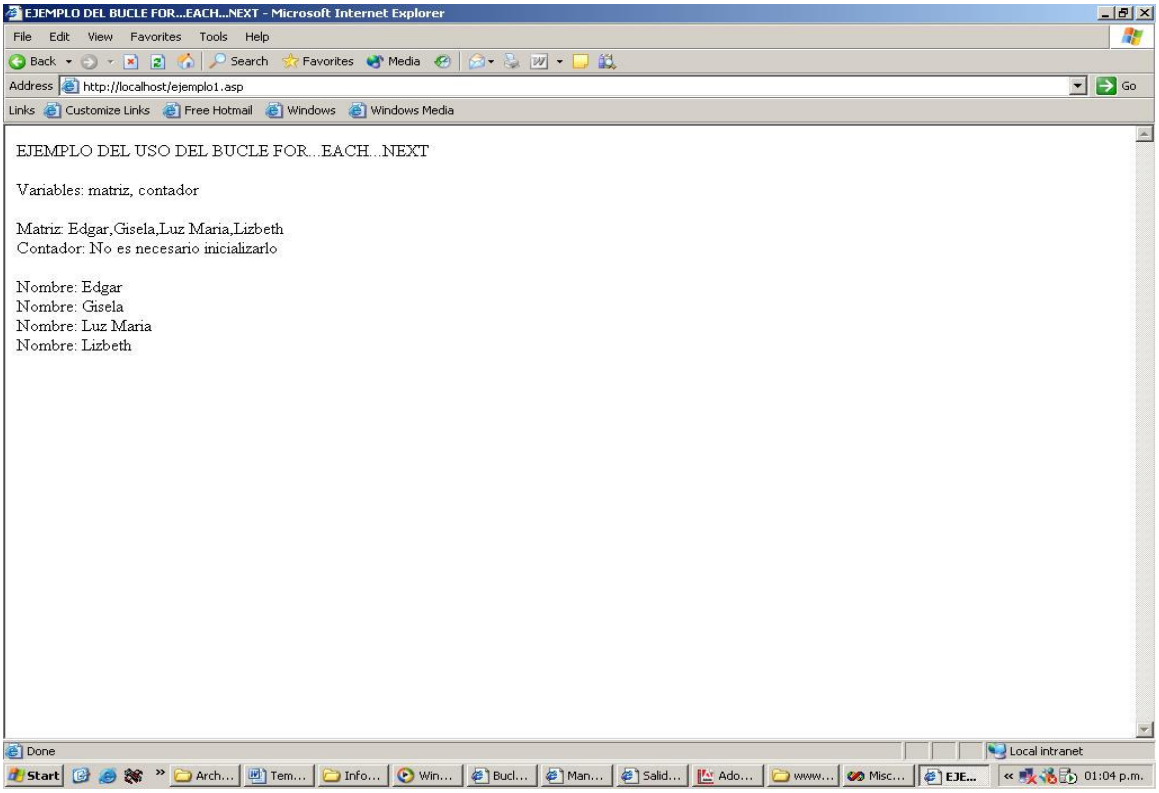


Figura 4.5. Bucle *For…Each…Next.*

Bucle Do…Loop.

Con este bucle podemos hacer que se repitan instrucciones dependiendo de una condición, es decir, permite hacer un numero de iteraciones no fijado de un inicio, se repite mientras dicha condición se cumpla.

Existen dos tipos de bucles *Do…Loop*, y estos pueden utilizarse de maneras distintas. Podemos hacer que un ciclo se repita mientras una condición se cumpla, para esto utilizamos la instrucción *While.* La otra opción es utilizando la instrucción *Until*, que sirve para crear una iteración hasta que una condición sea cierta, es parecida a la anterior. En estos bucles podemos utilizar la instrucción *Exit Do*, que permite salir del ciclo y nos ayuda a dejar la ejecución del bucle cuando sea necesario dejar de iterar en el ciclo.

La sintaxis del bucle *Do…While…Loop* es la siguiente:

```
Do While condicion
    Instruccion1
    Instruccion2
        .
        .
    InstruccionN
Loop
```

A continuación veremos un ejemplo de cómo utilizar este bucle:

```
<%
   Dim condicion, contador
   contador = 1
   condicion = 15
   Do While contador <= condicion
     Response.Write("Numero de Iteración: " & contador) & "<BR>"
     contador = contador + 1
   Loop
%>
```

[Vease Anexo A9]

La Figura 4.6 muestra el ejemplo anterior ejecutado en el explorador.

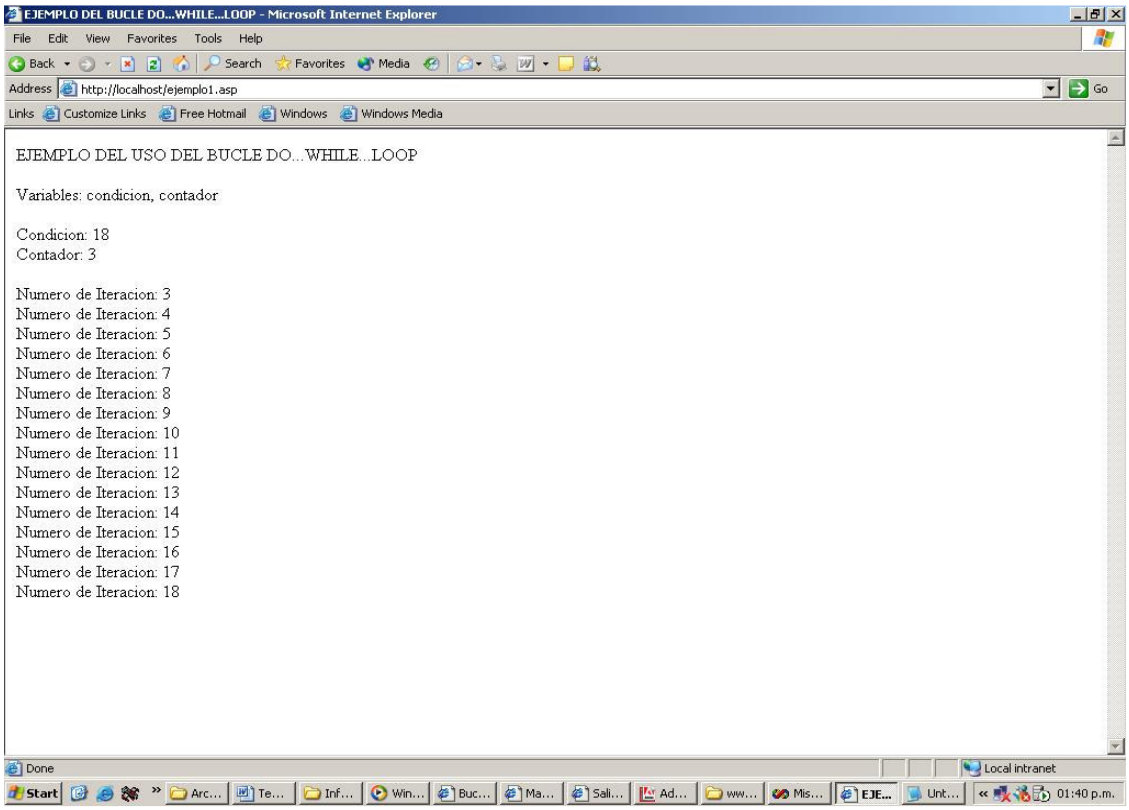


Figura 4.6. Bucle *Do…While…Loop*

La sintaxis del bucle *Do…Until…Loop* es parecida a la anterior y es de la forma siguiente:

```
Do Until condicion
    Instruccion1
    Instruccion2
        .
        .
    InstruccionN
Loop
```

El ejemplo del bucle *Do…Until…Loop* se muestra a continuación:

```
<%
   Dim condicion, meses
   meses = 1
   condicion = 12
   Do Until meses > condicion
     Response.Write("Estamos en el Mes Número: " & meses) & "<br>"
     meses = meses + 1
   Loop
%>
```

[Vease Anexo A10]

La Figura 4.7 muestra el programa anterior:

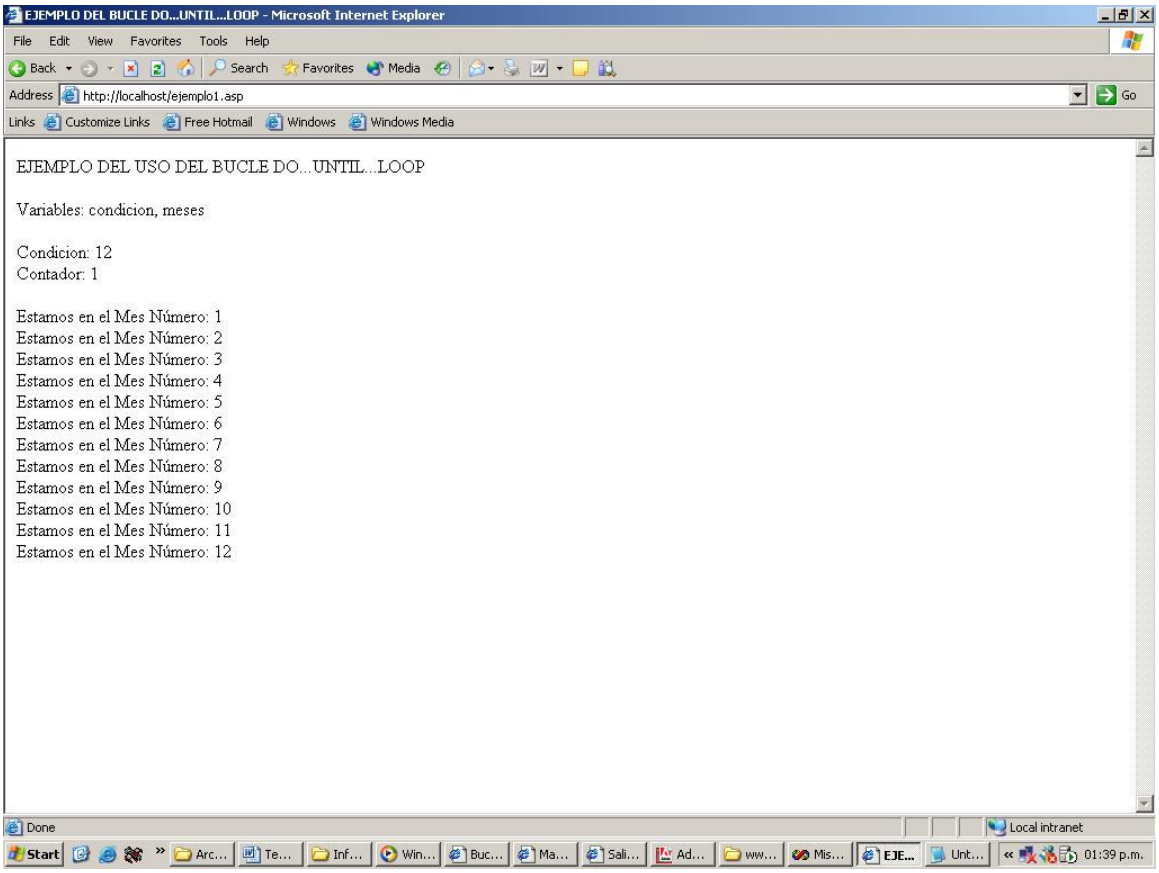


Figura 4.7 Bucle *Do…Until…Loop*

Otra forma de utilizar la instrucción *Until o While* es colocándola después de *Loop*. Con esto, indicamos que primero se debe ejecutar el trozo de código existente dentro del bucle *Do* y después verifique si se cumple la condición, si no se cumple, la ejecución del bucle se terminara y se saldrá del ciclo, por el contrario si la condición se cumple, las instrucción del trozo de código se vuelven a ejecutar hasta que ya no se cumpla la condición dada.

La sintaxis de esta forma de ejecutar el bucle en la forma anteriormente explicada es la siguiente:

Con la Instrucción UNTIL	*Con la Instrucción WHILE*
Do *Instruccion1* *Instruccion2* . . *InstruccionN* *Loop Until condición*	*Do* *Instruccion1* *Instruccion2* . . *InstruccionN* *Loop While condición*

Bucle While…Wend.

Este bucle tiene la misma estructura que el bucle *Do…While…Loop*. En la palabra *While* va la condición que queremos que se cumpla en el flujo del ciclo. Cuando la condición ya no se cumpla, el ciclo se terminara y el flujo del programa continuara después de la instrucción *Wend*.

A diferencia del bucle *Do…While*, en este tipo de bucle no se puede poner la condición de salida al final de la instrucción *Wend*. La sintaxis es mucho más sencilla y cómoda que la del bucle *Do…While*, aunque es muy poco flexible y menos funcional, sin embargo nos da una mejor claridad de nuestro programa. Es la siguiente:

```
While condición
   Instruccion1
   Instruccion2
      .
      .
   InstruccionN
Wend
```

Esta instrucción lo primero que hace es comprobar si la condición se cumple, si es así, entonces se ejecuta el código que se encuentra dentro del ciclo, cuando llega a *Wend*, regresa nuevamente a *While* y evalúa nuevamente la condición. Así continuara en el ciclo hasta que la condición ya no se cumpla y entonces la ejecución se saldrá del bucle.

El siguiente ejemplo muestra el uso del bucle *While*, en dicho ejemplo se realiza una iteración donde se suma uno a una variable y se imprime en pantalla. En cada ciclo comprueba si se cumple la condición especificada para ejecutar el código dentro del bucle. Una vez que la condición no se cumpla, el ciclo dejara de ejecutarse.

```
<%
   Dim condición, contador
   contador = 1979
   condicion = 2005
   While contador <= condición
            If contador = condición Then
                     Response.Write("Año en el que estoy viviendo actualmente: " & contador) & "<BR>"
            Else
                     Response.Write("Años en los que he vivido: " & contador) & "<BR>"
            End If
     contador = contador + 1
   Wend
%>
```

[Vease Anexo 11]

El ejemplo anterior se puede ver en la Figura 4.8.

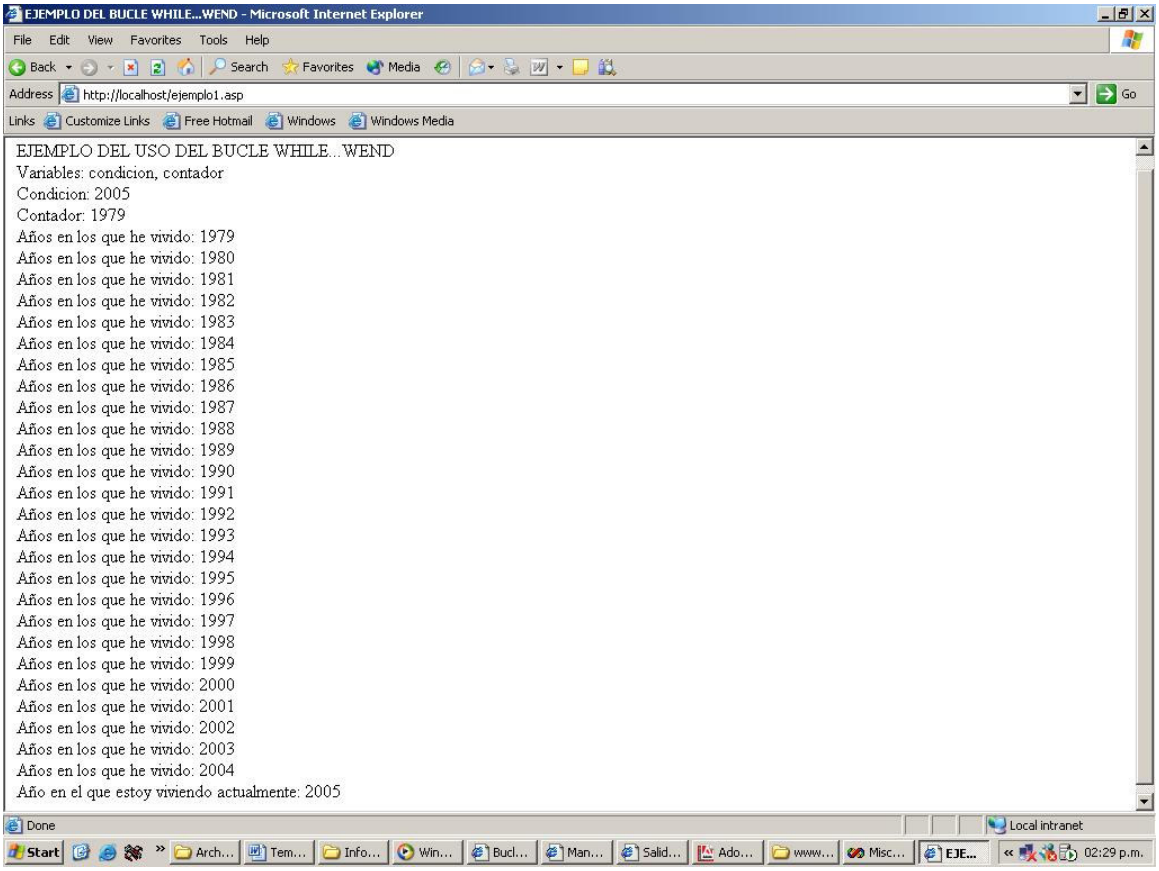


Figura 4.8 Bucle *While…Wend*

4.4. Cadenas.

El uso de cadenas es muy importante, para lo cual existen bastantes funciones para el manejo de estas, a continuación explicaremos las más usadas.

- ***Función len.*** Nos devuelve la longitud (numero de caracteres) de una cadena. La sintaxis de esta función es la siguiente:

  Len (Nombre_de_Cadena)

- ***Función Ubound.*** Obtiene el mayor subíndice disponible para la dimensión indicada de una cadena. La sintaxis de esta función es la siguiente:

  UBound (nombredecadena)

- ***Función split.*** Divide una cadena en varias usando un carácter separador. La sintaxis de esta función es la siguiente:

  Split (expresión, separador).

- ***Función mid.*** Devuelve una subcadena de otra, empezando por inicio y de longitud tamaño. La sintaxis de esta función es la siguiente:

  Mid (cadena, inicio, tamaño)

- ***Función trim.*** Elimina los espacios al inicio y final de una cadena. La sintaxis de esta función es la siguiente:

  Trim (Nombre_de_Cadena)

- ***Función instr.*** Busca una cadena dentro de otra y nos indica la posición en la que se encuentra. La sintaxis de esta función es la siguiente:
  Instr (cadena_donde_buscar, cadena_buscada)
- ***Función replace.*** Reemplaza la cadena1 por la cadena2 en el texto.
  Replace (expresión, búsqueda, reemplazar)

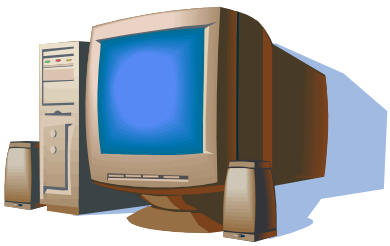
El siguiente ejemplo muestra el uso de las funciones aplicadas a una cadena:

```
<%
        Dim MiNombre, MiPais, MiEdad, MiApellido, i
        Dim FuncionLen, FuncionSplit, FuncionMid, FuncionTrim, FuncionInstr, FuncionReplace
        MiNombre = "Edgar Guerrero"
        MiPais = "Mexico!Brasil"
        MiEdad = " 26 "
        MiApellido = "Guerrero"
        FuncionLen = Len (MiNombre)
        FuncionSplit = Split (MiPais,"!")
        FuncionMid = Mid (MiNombre,7,14)
        FuncionTrim = Trim (MiEdad)
        FuncionInstr = Instr (MiNombre,MiApellido)
        FuncionReplace = Replace (MiPais,"!","@")
Response.Write("Ejemplo de la Función Len:") & "<BR>"
Response.Write("Tamaño de la cadena MiNombre: " & FuncionLen) & "<BR><BR>"
Response.Write("Ejemplo de la Función Split y de la función Ubound dentro de un bucle For:") & "<BR>"
For i = 0 To Ubound(FuncionSplit)
Response.Write("Muestra el contenido de la cadena MiPais ya subdividida: " & FuncionSplit(i)) & "<BR>"
Next
Response.Write("<BR>")
Response.Write("Ejemplo de la Función Mid:") & "<BR>"
Response.Write("De la cadena MiNombre muestra la palabra Guerrero: " & FuncionMid) & "<BR><BR>"
Response.Write("Ejemplo de la Función Trim:") & "<BR>"
Response.Write("De la cadena MiEdad quita los espacios al inicio y fin: " & FuncionTrim)&"<BR><BR>"
Response.Write("Ejemplo de la Función Instr:") & "<BR>"
Response.Write("En MiNombre busca MiApellido y da la posición: " & FuncionInstr) & "<BR><BR>"
Response.Write("Ejemplo de la Función Replace:") & "<BR>"
Response.Write("En MiPais cambia el carácter ! por el carácter @: " & FuncionReplace) & "<BR><BR>"
%>
```

[Vease Anexo 12]

CAPITULO 5

Procedimientos.



En este capitulo se mostrara las ventajas de usar funciones, procedimientos y librerías para la creación de nuestras paginas ASP.

5.1. Introducción.

Un procedimiento es una secuencia de comandos que se pueden ejecutar en cualquier momento en nuestro programa, haciendo referencia a su nombre. En VBScript existen dos tipos de procedimientos:

1. *Sub.*
2. *Function.*

La diferencia entre estos dos tipos es que el procedimiento *Function* devuelve un valor después de ser ejecutado, mientras que el procedimiento *Sub* no devuelve ningún valor. Esto nos indica que *Function* lo utilizaremos cuando necesitemos obtener algún valor o dato y *Sub* cuando no tengamos necesidad de devolver ningún valor después de ejecutar el procedimiento.

Los procedimientos nos sirven para agrupar instrucciones bajo un solo nombre y en un solo lugar y poder ejecutar dichas instrucciones varias veces desde diferentes partes de nuestro código, evitando con esto tener que escribirlas muchas veces y así repetir código, lo que conseguimos con los procedimientos es modularizar el código, si no se utilizaran en una aplicación un poco grande, esta se nos haría muy difícil de leer y darle mantenimiento, con lo cual seria intratable.

Si no existe ninguna función en nuestra pagina, el servidor procesa el archivo completo, cada vez que el explorador del cliente la solicita y los procedimientos se ejecutan solo cuando son llamados de alguna parte o evento de nuestro programa, no con el resto del código.

Para definir un procedimiento, podemos utilizar las etiquetas <*SCRIPT*> y </*SCRIPT*> indicando aquí el lenguaje que se va utilizar para crear el procedimiento. Estas etiquetas las utilizamos si queremos definir un procedimiento en un lenguaje distinto al principal, pero si lo queremos definir en el lenguaje principal que estemos usando, lo hacemos dentro de las etiquetas <% y %>.

Los procedimientos, ya sea del tipo *Sub* o *Function* pueden tener parámetros, los parámetros son datos o variables que podemos pasar a los procedimientos. Para declarar un procedimiento con parámetros se colocan entre paréntesis después del nombre del procedimiento y separados por comas, si no se necesita mandar ningún dato, solo se colocan los paréntesis vacíos:

nombre_procedimiento(parametro1, parametro2, … , parametroN)

El nombre del procedimiento puede ser cualquier nombre valido para el lenguaje, es decir, que no sean palabras reservadas del lenguaje. La restricción que tenemos en el lenguaje es que no podemos anidar procedimientos.

Se pueden definir los procedimientos en el mismo archivo .asp que los utiliza o ponerlos en otro archivo .asp y para ejecutarlos, utilizar la directiva *#include* para ejecutarlos en nuestra pagina o en la que sean necesario utilizarlos, esta opción son las librerías, las cuales veremos mas adelante.

5.2. Procedimiento Sub.

Este tipo de procedimientos se caracteriza por no regresar ningún valor después de ser ejecutado, esto significa que lo que nos sirve por las acciones que se realicen con este procedimiento y no el resultado de estas. Por ejemplo podemos utilizar un procedimiento *Sub* para inicializar variables o interactuar con el usuario, entre otras.

Los procedimientos *Sub* se pueden definir en cualquier momento de nuestro programa y la sintaxis para hacerlo es la siguiente:

```
Sub Nombre_procedimientoSub()
    Instruccion1
    Instruccion2
    Instruccion3
            .
            .
    InstruccionN
End Sub
```

Podemos añadir las cláusulas *Public* o *Private* en la declaración del procedimiento y dicha cláusula nos definirá el alcance del procedimiento. El siguiente ejemplo del procedimiento *Sub* imprime el resultado de realizar una división entre dos números:

```
Sub DivisionSub()
        Dim resultado, valor1, valor2
        valor1 = 2027
        valor2 = 23
        resultado = valor1 / valor2
        Response.Write("<B>EJEMPLO DEL PROCEDIMIENTO SUB</B>")&"<BR><BR>"
        Response.Write("<LI>Variables: resultado, valor1, valor2")&"<BR>"
        Response.Write("    valor1: " & valor1)&"<BR>"
        Response.Write("    valor2: " & valor2)&"<BR><BR>"
        Response.Write("<LI>Se realiza la siguiente operación:")&"<BR>"
        Response.Write("    resultado = valor1 / valor2")&"<BR><BR>"
        Response.Write "<LI>El resultado de la división es: " & resultado
End Sub
```

En VBScript existe la instrucción *Exit Sub* que sirve para salir de un procedimiento *Sub* en el momento en que lo deseemos, sin necesidad de terminar de ejecutar todo el procedimiento. Con esta instrucción podemos hacer que nuestro procedimiento tenga varias formas de ser terminado.

5.3. Procedimiento Function.

Este tipo de procedimiento nos devuelve un valor después de ser ejecutado. Aunque una función puede hacer todo lo que hace un procedimiento *Sub*, la diferencia y finalidad de *Function* es devolver un valor.
Para declarar un procedimiento *Function* utilizamos la misma sintaxis que usamos para *Sub*, simplemente cambiamos *Sub* y *End Sub* por *Function* y *End Function*:

```
Function Nombre_procedimientoFunction()
    Instruccion1
    Instruccion2
    Instruccion3
            .
```

```
        .
    Nombre_procedimientoFunction = valor_devuelto
        .
        .
    InstruccionN
End Function
```

Como se puede ver en la sintaxis, utilizamos el nombre de la función para devolver un valor, es decir, asignamos al nombre de la función el valor que se va a devolver, lo cual se puede hacer en cualquier parte de la función. Al igual que en el anterior procedimiento, podemos añadir las cláusulas *Public* o *Private* en la declaración del procedimiento. El siguiente ejemplo es equivalente al visto en el procedimiento *Sub* solo que se realiza con una función. En este caso en lugar de imprimir el resultado, se devuelve el valor:

```
Function DivisionFunction()
        Dim resultado, valor1, valor2
        valor1 = 2027
        valor2 = 23
        resultado = valor1 / valor2
        DivisionFunction = resultado
End Function
```

En este procedimiento al igual que con el anterior, existe una forma de terminarlo en el momento que deseemos hacerlo, utilizando la instrucción *Exit Function*.

5.4. Llamada a un Procedimiento.

El llamado a un procedimiento del tipo *Sub* lo podemos hacer de la siguiente manera:

1. Escribiendo solamente el nombre junto con los parámetros (si es que tiene) separados por comas: *nombre_del_procedimiento parametro1, parametro2,…*
2. Utilizando la instrucción Call y escribiendo el nombre del procedimiento escribiendo los parámetros entre paréntesis (si es que tiene, si no se quedan vacíos: *Call nombre_del_procedimiento(parametro1, parametro2,..)*

El llamado a un procedimiento del tipo *Function* lo podemos hacer de la siguiente manera:

1. Asignando la llamada del procedimiento a una variable, ya que la función nos devolverá un valor al ser ejecutada: *resultado = nombre_del_procedimiento()*
2. Se puede utilizar también para este procedimiento la instrucción *Call*, sin embargo esta instrucción ignora el valor devuelto por el procedimiento, por lo cual no nos será de mucha utilidad.

Se puede llamar a un procedimiento antes de que sea declarado siempre y cuando este posteriormente declarado en nuestro programa que se va a ejecutar. Veamos un ejemplo del uso y llamado de los dos tipos de procedimientos vistos anteriormente:

```
<%
Dim ValorDevuelto
Sub DivisionSub()
        Dim resultado, valor1, valor2
        valor1 = 2027
```

```
        valor2 = 23
        resultado = valor1 / valor2
        Response.Write("<B>EJEMPLO DEL PROCEDIMIENTO SUB</B>")&"<BR><BR>"
        Response.Write("<LI>Variables: resultado, valor1, valor2")&"<BR>"
        Response.Write("    valor1: " & valor1)&"<BR>"
        Response.Write("    valor2: " & valor2)&"<BR><BR>"
        Response.Write("<LI>Se realiza la siguiente operación:")&"<BR>"
        Response.Write("    resultado = valor1 / valor2")&"<BR><BR>"
        Response.Write "<LI>El resultado de la división es: " & resultado
End Sub
Function DivisionFunction()
        Dim resultado, valor1, valor2
        valor1 = 2027
        valor2 = 23
        resultado = valor1 / valor2
        DivisionFunction = resultado
End Function
ValorDevuelto = DivisionFunction()
%>
<font face="arial" size="2" color="black"><li>El resultado de la división es: <%=ValorDevuelto%></font>
```

[Vease Anexo A13]

La ejecución del ejemplo anterior se aprecia en la figura 5.1:

5.5. Librerías.

El uso de librerías es muy útil, nos permiten agrupar varias funciones y variables en un mismo fichero, de forma que después podemos incluir esta librería en distintas páginas y disponer de esas funciones fácilmente.

Para hacer uso de estos ficheros, utilizamos la directiva *#include* en la pagina donde queremos utilizar algún procedimiento de esa librería y de este modo ejecutarla en la pagina actual.

Vamos a crear una página que solo contenga procedimientos para después utilizarla como librería:

```
<%Sub CabeceraPagina()%>
  <table width="100%">
  <tr>
  <td align="right">
  <font face="arial" size="2" color="black">Utilización de librerías en ASP.</font>
  </td>
  </tr>
  </table>
  <hr>
<%End Sub
 Function CuerpoPagina()
    CuerpoPagina = Date()
 End Function
 Sub PiePagina()%>
  <hr>
  <table width="100%">
  <tr>
  <td align="left">
  <font face="arial" size="2" color="black">Ejercicio Sencillo de Utilerías.</font>
  </td>
  <td align="right">
```

```
  <font face="arial" size="2" color="black">Autor: Edgar Antonio Guerrero González</font>
  </td>
  </tr>
  </table>
<% End Sub%>
```

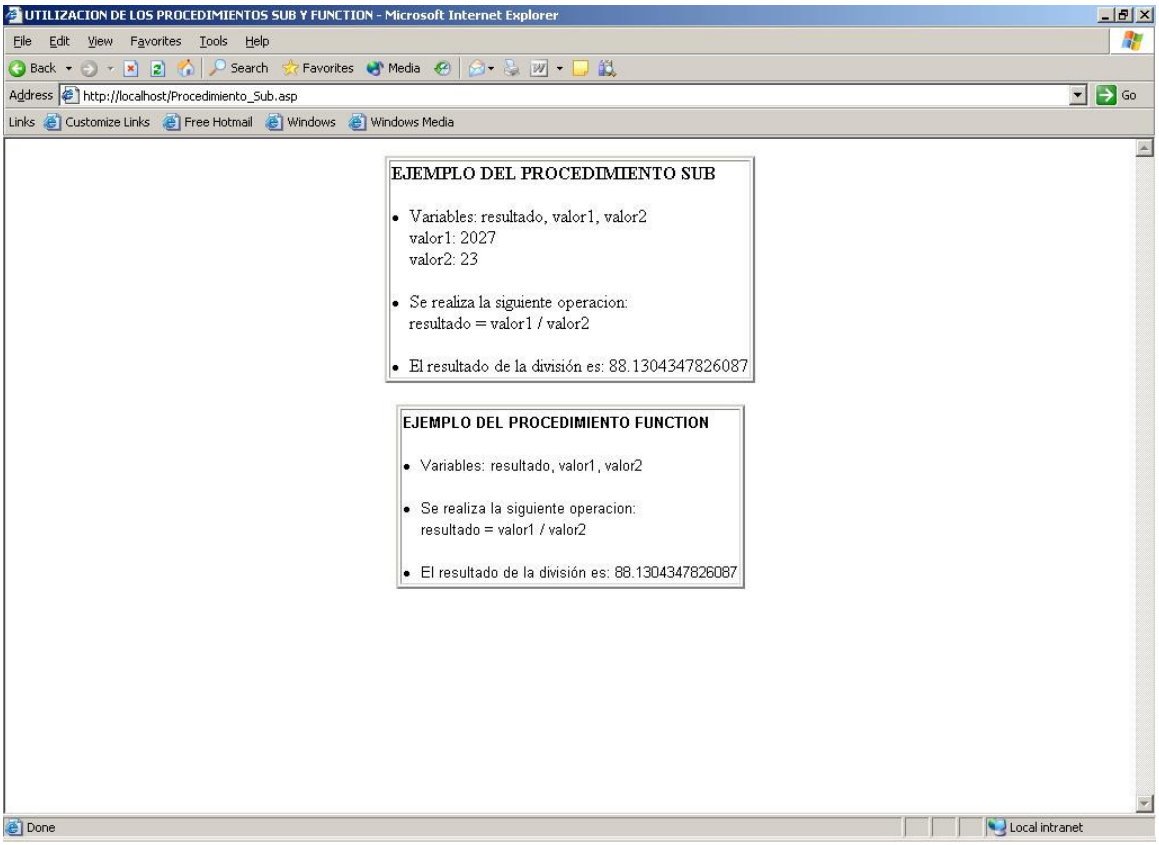


Figura 5.1 Procedimientos *Sub* y *Function*.

En la página siguiente vamos a utilizar los procedimientos declarados en el archivo anterior:

```
<%Dim resultado%>
<!-- #include file="ejemplo_libreria.asp" -->
<%Call CabeceraPagina()
 resultado = CuerpoPagina()%>

<font face="arial" size="2" color="blue"><li>Esta página ejecuta Procedimientos que se encuentran declarados en
otro archivo y que</font>

<font face="arial" size="2" color="blue">    se muestran en el encabezado y pie de esta
página, así como la fecha del día hoy.</font>

<font face="arial" size="2" color="blue"><li>Hoy estamos a: <%=resultado%></font>
<%Call PiePagina()%>
```

[Vease Anexo A14]

Al hacer lo anterior puede ejecutar los procedimientos existentes en ese archivo como si estuvieran declarados en la misma página. De esta manera los procedimientos *CabeceraPagina(), PiePagina() y CuerpoPagina()* son ejecutados y mostrados en el

explorador sin estar físicamente en la página que los llama. Si necesitáramos utilizar estos procedimientos en otras páginas, solo tendríamos que utilizar como librería el archivo que los tiene y así ya no tenemos la necesidad de repetir el mimo código en cada página donde lo ocupemos.

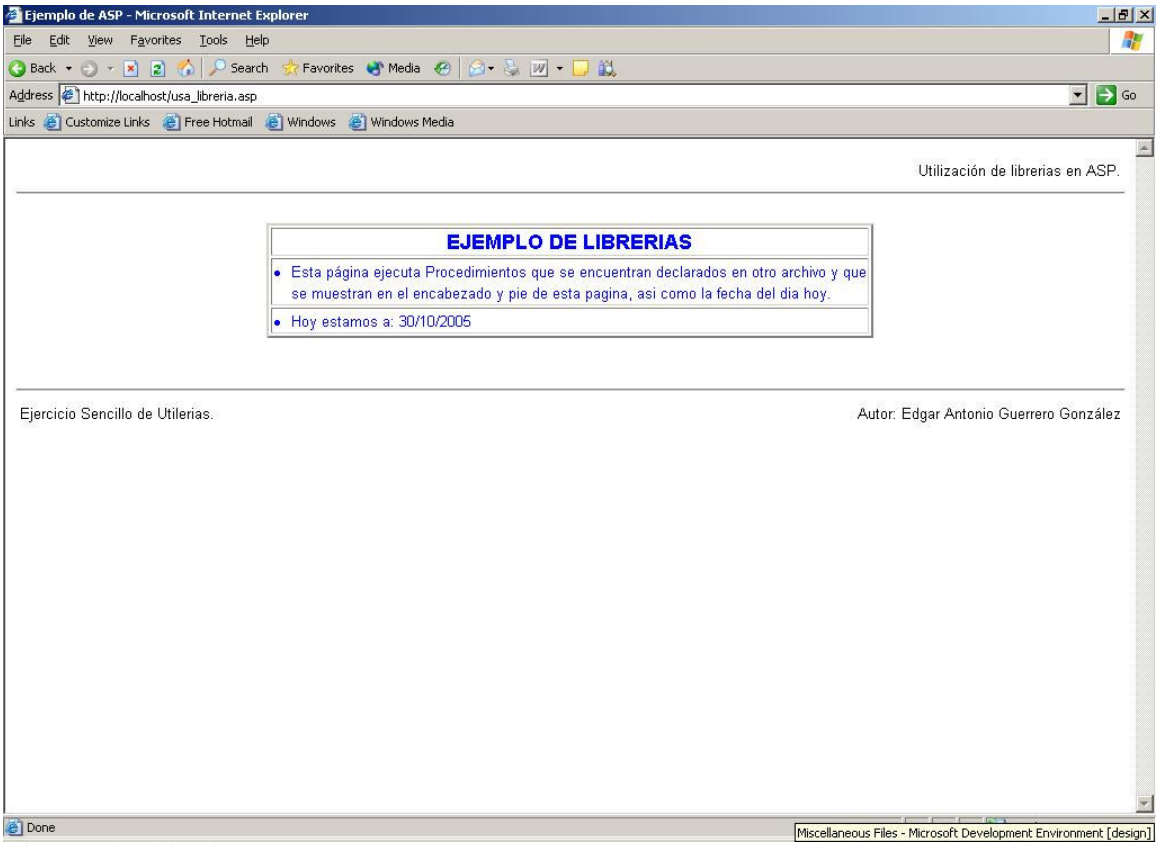


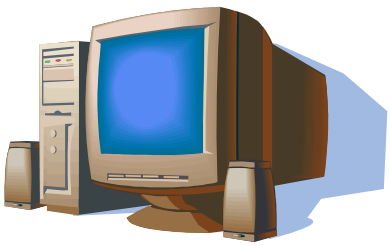
Figura 5.2 Utilización de Librerías en ASP.

En la Figura 5.2 se muestra como la segunda página utiliza como librería la primera al hacer referencia a ella utilizando la instrucción:

<!-- #include file="ejemplo_libreria.asp" -->

CAPITULO 6

Procesamiento de Formularios.



En este capitulo se hablara de la forma de manejar formularios para procesar la información que el usuario ha introducido, así como los distintos métodos para hacerlo.

6.1. Introducción.

Los formularios de HTML, son el método más común y utilizado para obtener información en la Web. Son estructuras lógicas de un documento HTML que pueden contener distintos controles donde el usuario puede manejar e introducir información y que son elementos de interfaz de usuario en una pagina Web. ASP no sustituye a los formularios, sino que los mejora y los hace más fácil de implementar y administrar.

Algunos ejemplos de los controles mencionados son los cuadros de texto, listas, botones, checkboxes, radio botones, etc., y están contenidos dentro de los formularios en una pagina, para enviar información a un servidor Web.

Para hacer un formulario, se utiliza la etiqueta <FORM> y utiliza los siguientes parámetros importantes en su estructura:

- *ACTION:* Indica la URL a donde hay que enviar los datos contenidos en el formulario. En nuestro caso indica la pagina ASP a donde se enviaran los datos.
- *METHOD:* Indica el método que va a utilizar el formulario para enviar la información. Existen dos métodos posibles: el método *Get* y el método *Post,* los cuales veremos en los siguientes puntos.

La sintaxis de declaración de un formulario es como se muestra a continuación:

<form action=*url_destino* id=*nom_formulario* name= *nom_formulario* method=*tipo_metodo*>
    Estructuras para enviar información
</form>

Para los formularios existe también un botón de tipo *Submit*, sin tener que programar la acción de dicho botón y que al ser pulsado envía toda la información contenida en el formulario a la URL definida en el action.

Existen tres formas de recopilar información de un formulario HTML con ASP:

1. Con un archivo .html estático, en donde su formulario envíe la información a un archivo .asp
2. con un archivo .asp que tenga un formulario que envíe la información a otro archivo .asp
3. con un archivo .asp que tenga un formulario que envíe la información a si mismo.

Los formularios envían información al servidor y este la procesa y la envía a la URL indicada en el parámetro *ACTION* y dentro de esta es utilizada para realizar lo que se desea, desde simplemente mostrarla en el explorador, hasta guardarla en una base de datos.

6.2. Método GET.

Como ya se explico al inicio del capitulo los datos de un formulario se envía mediante el método indicado en el atributo *METHOD* de la etiqueta *FORM*, y uno de ellos es el método *GET*.

Este método sirve para enviar los datos de un formulario a través de la URL, es decir por parámetro y dichos datos pueden verse en ella.

La forma de utilizar este método se muestra en el siguiente ejemplo:

```
<FORM ACTION="metodo_get.asp" METHOD="GET">
Introduzca su nombre:<INPUT TYPE="text" NAME="nombre"><BR>
Introduzca sus apellidos:<INPUT TYPE="text" NAME="apellidos"><BR>
<INPUT TYPE="submit" VALUE="Enviar">
</FORM>
```

Al pasar dichos datos mediante este método, se verían en la barra de direcciones del explorador de la siguiente manera:

http://localhost/metodo_get.asp?nombre=Edgar+Antonio&apellidos=Guerrero+Gonzalez

También podemos enviar datos por parámetro generándolo de forma "manual", es decir, después de la dirección URL se pone el signo "?" y a continuación todos los parámetros con su respectivo valor separados por el símbolo *&*. Ejemplo:

http://localhost/metodo_get.asp?edad=26&nacionalidad=Mexicana

Ninguna de las dos formas anteriores son las más recomendadas para enviar datos de un formulario, ya que tiene 2 inconvenientes:

1. Una dirección URL esta limitada en cuanto al número de caracteres que puede tener, solo permite 1024, por este motivo si nuestro formulario tiene muchos datos, estos serán cortados y no llegaría toda la información enviada.
2. Cuando se envían datos a través de la URL, esta viaja en la cabecera del mensaje del protocolo http, es decir en la barra de direcciones del explorador. Esto implica qué son muy vulnerables a ser leídos por terceras personas, aunque se usen protocolos de seguridad, esto hace que el envío de información sea insegura.

Por lo anterior existe otro método más confiable para el envío de datos de un formulario, el cual veremos en el siguiente punto.

6.3. Método POST.

El método POST también sirve para enviar datos, y a diferencia del método GET, con este, los datos enviados no se ven en la barra del explorador, si no que los datos viajan dentro del cuerpo del mensaje de protocolo http, con lo cual es posible codificar dichos datos con certificados de seguridad como SSL, por lo que nuestra información esta segura, además no existe limite en cuanto al numero de caracteres que se pueden enviar. Por este motivo es el método mas utilizado para el envío de formularios.

A continuación veremos la forma de utilizarlo, que es igual que con el método get, solo se indica en el parámetro *METHOD* la forma que se va a utilizar:

```
<FORM ACTION="metodo_post.asp" METHOD="POST">
Introduzca su nombre:<INPUT TYPE="text" NAME="nombre"><BR>
Introduzca sus apellidos:<INPUT TYPE="text" NAME="apellidos"><BR>
```

```
<INPUT TYPE="submit" VALUE="Enviar">
</FORM>
```

6.4. Envío y Recepción de Datos.

Como ya vimos podemos enviar información por formularios de manera sencilla, permitiéndonos de esta manera procesar la información que el usuario ha introducido.

Dependiendo de como enviemos la información de nuestro formulario, utilizaremos diferentes formas para recibirla para posteriormente procesarla. Si usamos el método *GET,* utilizamos la propiedad QueryString, que es en realidad un colección, es decir una lista de valores (algo parecido a un Array, pero donde cada elemento tiene un nombre que lo identifica). Si usamos el método *POST*, utilizaremos la propiedad *Form*, que al igual que la anterior, también es una colección de elementos.

Las dos propiedades mencionadas pertenecen al objeto *Request* (el cual explicaremos en el siguiente capitulo) y la forma para recuperar información usando estas colecciones es poniendo detrás de estas el nombre del elemento que deseamos obtener entre comillas y paréntesis, la sintaxis es la siguiente:

- *Colección QueryString:* Request.QueryString("nombre_del_elemento")
- *Colección Form:* Request.Form("nombre_del_elemento")

A continuación veremos un Ejemplo de la utilización de un formulario usando el método *POST*:

```
<form action="mostrar_formulario.asp" id="form1" name="form1" method="post">
<center>
<table width="50%" align="middle" border="2">
<tr>
<td width="50%" align="left">
<font face="arial" color="black" size="2">Nombre :</font>
</td>
<td width="50%" align="left">
<input id="txtnombre" name="txtnombre" >
</td>
</tr>
<tr>
<td width="50%" align="left">
<font face="arial" color="black" size="2">Edad :</font>
</td>
<td width="50%" align="left">
<input id="txtedad" name="txtedad" size="2" maxlength="2">
</td>
</tr>
</table>
<br>
<input type="submit" id="btnaceptar" name="btnaceptar" value="Enviar Información">
</center>
</form>
```

[Vease Anexo A15]

En el formulario anterior el usuario puede indicar su nombre, edad, sexo y país, e incluye

un botón de tipo submit para enviar la información(es decir el nombre y la edad que el usuario haya capturado en las cajas de texto, el sexo que haya indicado en los radio botones y el país que haya seleccionado en el combo) al servidor y este a su vez a la pagina mostrar_formulario.asp indicada en el action del formulario; la figura 6.1 muestra el formulario procesado en el explorador:

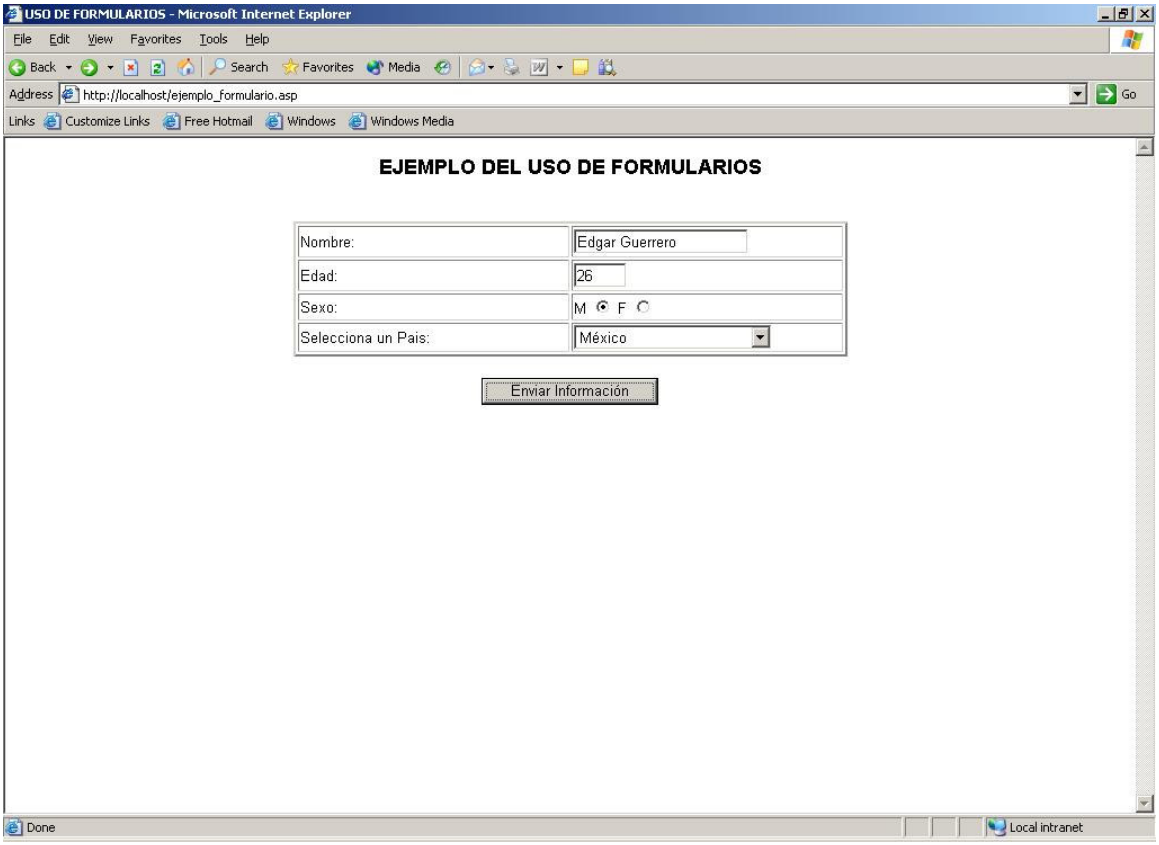


Figura 6.1 Ejemplo de Formulario.

Para utilizar el método *GET*, solo lo indicamos en el parámetro *METHOD* del formulario de la página.

6.5. Validar los Datos del Formulario.

Un formulario bien diseñado suele incluir una secuencia de comandos del cliente (programas scripts) que valida los datos proporcionados por el usuario antes de enviar la información al servidor. Estos scripts de validación pueden comprobar si el usuario un numero valido o si un cuadro de texto esta en blanco, entre otras, antes de enviar información al servidor que podría no ser valida.

Lo más recomendable es realizar en el cliente todas las comprobaciones que sean posibles, además de preguntar antes al usuario por los errores; la validación en el cliente mejora el tiempo de respuesta, reduce la carga del servidor y libera ancho de banda para otras aplicaciones.

Lo anterior se hace declarando procedimientos dentro de nuestra pagina utilizando la etiqueta <*SCRIPT LANGUAJE="lenguaje_a_utilizar"*> y declarando el lenguaje que se va a utilizar para ello. La sintaxis para hacerlo es la siguiente:

Utilizando JAVASCRIPT	***Utilizando VISUAL BASIC SCRIPT***
<*script languaje*="javascript"> *Instruccion1* *Instruccion2* . . *InstruccionN* </*script*>	<*script languaje*="vbscript"> *Instruccion1* *Instruccion2* . . *InstruccionN* </*script*>

Veamos un ejemplo de validación utilizando el formulario del ejemplo del punto anterior y usando *JAVASCRIPT* como lenguaje para hacerlo:

```
<script language="JavaScript">
        function Validar_Datos()
        {
                if(document.form1.txtNombre.value == "")
                {
                        alert("Debe indicar su nombre completo.")
                        document.form1.txtNombre.focus()
                        return(false)
                }
                if(document.form1.txtEdad.value == "")
                {
                        alert("Debe indicar su edad.")
                        document.form1.txtEdad.focus()
                        return(false)
                }

                var cadena = 0
                for(i=0;i<document.form1.elements.length;i++)
                {
                        if(document.form1.elements[i].type == "radio")
                        {
                                if(document.form1.elements[i].checked == true)
                                {
                                        cadena = 1
                                }
                        }
                }
                if(cadena == 0)
                {
                        alert("Debe indicar su sexo.")
                        return(false)
                }
                if(document.form1.cmbPais.value == "0")
                {
                        alert("Debe seleccionar su país de origen.")
                        document.form1.cmbPais.focus()
                        return(false)
                }
                if(confirm("¿Todos los datos son correctos?") == true)
```

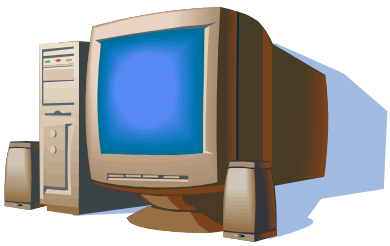
```
                {
                        return(true)
                }
                else
                {
                        return(false)
                }
        }

        function Valida_Numeros(dato)
        {
          var cadenaOK = "0123456789";
          var cadenaStr = dato.value;
          var Valido = true;
          for(i=0;i<cadenaStr.length;i++)
          {
            ch = cadenaStr.charAt(i);
            for(j=0;j<cadenaOK.length;j++)
              if(ch == cadenaOK.charAt(j))
                break;
              if(j == cadenaOK.length)
              {
                        Valido = false;
                        break;
              }
          }
          if (!Valido)
          {
                alert("Escriba sólo dígitos en este campo.");
                dato.value="";
                dato.focus();
                return (false);
          }
        }
</script>
```

[Vease Anexo A16]

Podemos ver que con estas validaciones, no se permitirá enviar el formulario con los datos vacíos, si no hasta que se llenen los datos, que en el campo edad se introduzcan solo números enteros (en este campo utilizamos el evento *onblur* para realizar esta validación) y que el usuario confirme que dichos datos están correctos. Para este caso, en donde estamos utilizando el botón de tipo *submit*, debemos utilizar el evento *onsubmit* en el formulario, ya que si usamos otro, como por ejemplo el *onclick*, la información seria enviada sin importar dichas validaciones. Observemos que con este script, todas las validaciones se hacen en el lado del cliente, liberando al servidor de esta responsabilidad.

CAPITULO 7

Objetos Integrados de ASP.



En este capitulo se mostraran los diferentes tipos de objetos que utiliza ASP para el manejo de la información de la paginas.

7.1. Introducción.

La mayor parte de las aplicaciones Web requieren hacer intercambio de información entre clientes y servidores. Al diseñar un sitio Web resulta necesario utilizar elementos para enviar y recoger información. ¿Como podemos utilizar en las paginas ASP la información que los usuarios proporcionan? ASP proporciona un método sencillo para hacerlo porque encapsula los detalles de la obtención de datos en la funcionalidad de los *objetos integrados.*

El lenguaje ASP es una plataforma basada en objetos, esto significa que ofrece un conjunto de objetos para ampliar la funcionalidad del lenguaje que estemos utilizando (por ejemplo de VBScript).

La plataforma invoca a los denominados objetos los cuales son módulos incorporados al lenguaje que permiten el desarrollo de tareas específicas. Estos objetos realizan de una manera sencilla toda una serie de acciones, a partir de una llamada al objeto este realizará la tarea requerida. En cierta forma, estos objetos nos ahorran el tener que hacer largos programas para operaciones sencillas.

Como todo objeto del mundo real, los objetos de ASP tienen sus propiedades que los definen, realizan un cierto número de funciones o métodos y son capaces de responder de cierta forma ante algunos eventos.

En ASP podemos encontrar cinco objetos, los cuales como se menciono anteriormente, contienen los métodos, propiedades y eventos para su funcionamiento. Estos cinco objetos son:

- ***Application.-*** Devuelve una referencia del objeto *Application*, con lo cual se puede compartir información entre todos los clientes de una aplicación.
- ***Request.-*** Recibe información del cliente, tanto de la información que el proporciona como de los datos ocultos de su entorno (dirección IP, tipo de explorador, etc.).
- ***Response.-*** Envía información al cliente, tanto código HTML como información de ficheros, cabeceras, etc.
- ***Session.-*** Permite almacenar información referente a un cliente concreto, ya que abarca el ámbito de una sesión.
- ***Server.-*** Permite conocer el entorno de ejecución de ASP.

Estos cinco objetos son implícitos y son fijos de ASP, y son la base de toda la plataforma.

Existen sin embargo otro tipo de objetos, llamados componentes de ASP, que sirven para ampliar la funcionalidad de nuestras páginas y son instalados por defecto con el lenguaje. Estos componentes son utilizados para acceder a datos almacenados en una base de datos, por lo cual no se mostraran en este capitulo, los trataremos en el capitulo 10 de Base de Datos.

En este trabajo mostraremos los métodos y propiedades más frecuentemente utilizados de los cinco objetos descritos anteriormente.

7.2. Objeto Application.

Este objeto se utiliza para compartir información entre todos los usuarios de una aplicación (entendemos por una aplicación ASP todos los archivos .asp de un directorio virtual y sus subdirectorios), por lo tanto es único para todo el sitio Web. Podemos utilizarlo para guardar variables y datos que podemos recuperar desde nuestras páginas y todos los usuarios compartirán el mismo objeto.

Como ya vimos en el capitulo de variables, podemos crearlas utilizando la palabra reservada *Dim*, pero estas variables solo pueden ser utilizadas dentro de nuestra página o procedimiento o función donde la hayamos creado, sin embargo en ocasiones necesitaremos crear variables globales que sean visibles en todas partes, para eso utilizamos las variables de aplicación, las cuales consisten en toda la información que guardamos en dicho objeto y se escriben de la siguiente forma:

Application("nom_variable")

Ahora veamos un sencillo ejemplo del uso de variables de aplicación:

```
<%
        Application("Nombre") = "Edgar Guerrero"
        Application("Apellidos") = "Guerrero González"
        Application("Edad") = 26
        Application("n_visitas") = Application("n_visitas") + 1
%>
  <tr>
   <td>
    <font face="arial" color="black" size="2"><b>eres el visitante número:
<%=application("n_visitas")%></b></font>
   </td>
  </tr>
 <table width="50%" align="middle" border="2">
  <tr>
   <td width="50%" align="left">
    <font face="arial" color="black" size="2">Nombre(s):</font>
   </td>
   <td width="50%" align="left">
    <input id="txtnombre" name="txtnombre" value="<%=Application("Nombre")%>">
   </td>
  </tr>
  <tr>
   <td width="50%" align="left">
    <font face="arial" color="black" size="2">Apellidos:</font>
   </td>
   <td width="50%" align="left">
    <input id="txtapellidos" name="txtapellidos" value="<%=Application("Apellidos")%>">
   </td>
  </tr>
  <tr>
   <td width="50%" align="left">
    <font face="arial" color="black" size="2">Edad:</font>
   </td>
   <td width="50%" align="left">
    <input id="txtedad" name="txtedad" size="2" maxlength="2" value="<%=Application("Edad")%>">
   </td>
 </table>
```

[Vease Anexo A17]

Como podemos ver en este ejemplo, las variables de aplicación creadas, pueden ser utilizadas en cualquier parte de nuestra página y la que indica el número de visitante es modificada cada vez que un usuario entra a nuestra pagina. Si pasáramos a otra, en esa también podríamos utilizar dichas variables, veámosla ejecutada en el explorador (vease figura 7.1).

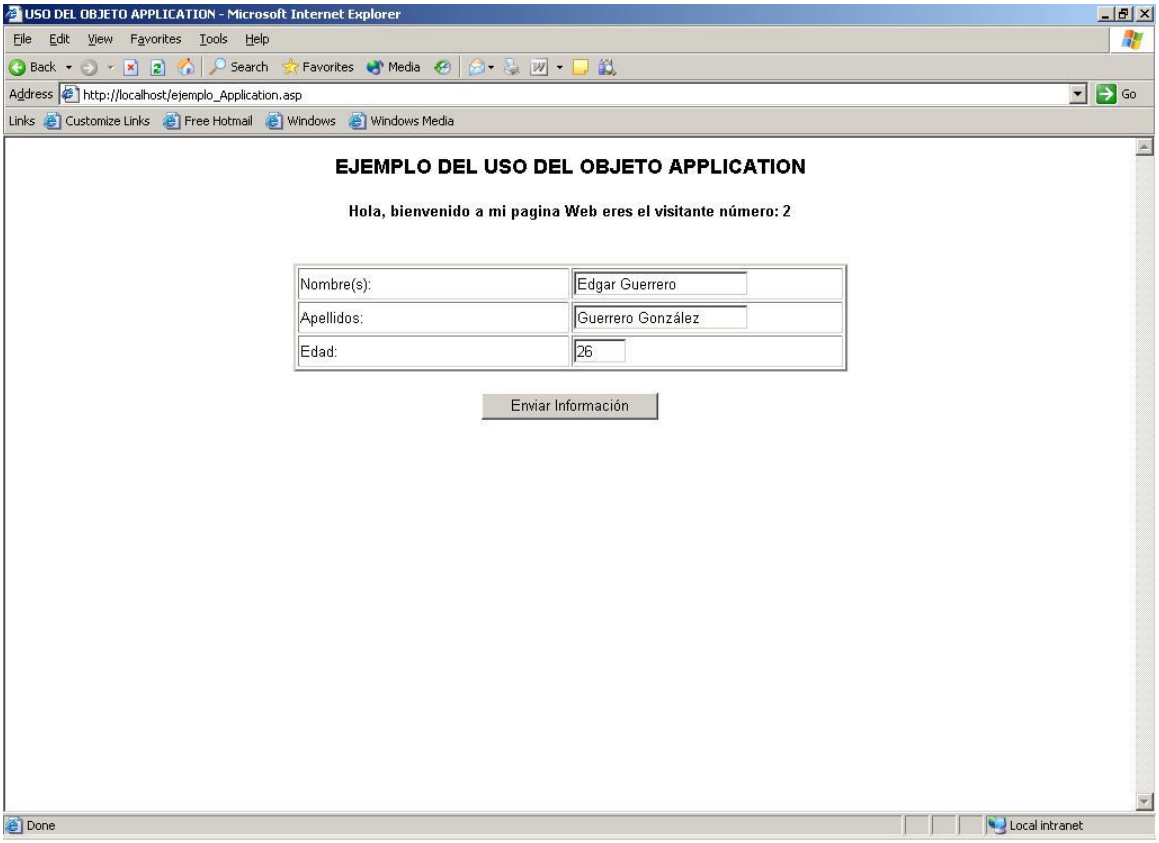


Figura 7.1 Ejemplo del Objeto *Application*.

El uso de las variables de aplicación tiene un problema: ya que el contenido del objeto *Application* es global y compartido por todos los usuarios del sitio, puede existir un problema de *concurrencia*. Por ejemplo si en un proceso de alguna pagina, se permite la modificación del valor de una variable de aplicación y dos o mas usuarios entran al mismo tiempo y la modifican, no se puede garantizar que utilicen el mismo valor o que se haga la modificación. Por lo tanto tenemos que garantizar que los acceso para modificar una variable de aplicación son únicos (es decir atómicos). Para ello existen los métodos *Lock (cerrar)* y *Unlock (abrir)* para asegurar la integridad del mismo, con lo cual el uso de la variable de aplicación se restringe en ese momento y así ningún otro usuario pueda modificarla al mismo tiempo, si no hasta que el usuario termine de utilizarla.

Método Lock.

El método *Lock* asegura que sólo un usuario puede modificar o tener acceso al objeto *Application* al mismo tiempo. La Sintaxis de declaración es la siguiente:

Application.Lock

Método Unlock.

El método *Unlock* desbloquea el objeto *Application* para que pueda ser modificado por otro usuario después de haber sido bloqueado mediante el método *Lock*. Si no se llama a este método de forma explícita, el servidor Web desbloquea el objeto Application cuando la página termina de ejecutarse o transcurre su tiempo de espera. La Sintaxis de declaración es la siguiente:

```
Application.Unlock
```

Veamos en el siguiente ejemplo donde utilizaremos la variable anterior con la cual modificamos el número de visitas a nuestro sitio:

```
Application.Lock
Application("n_visitas") = Application("n_visitas") + 1
Application.Unlock
```

De esta forma, cualquier otro proceso o usuario que quiera escribir en la variable de aplicación quedara en espera, hasta que sea llamado el método *Unlock*, y con esto garantizamos que utilizara el valor correcto.

Podemos ver entonces, que el objeto *Application* es una buena forma para almacenar datos que utilicemos muchas veces en el sitio Web. El acceso a la memoria del servidor Web para leer una variable de aplicación es muchísimo más rápido que obtener los datos de otro origen.

Sin embargo también tiene sus desventajas este objeto ya que como se menciono, se guarda en la memoria del servidor Web, esto significa que si lo cargamos demasiado, el servidor comenzará a disminuir su rendimiento.

El objeto *Application* es uno de los objetos intrínsecos de ASP más conocidos y nombrados, y a la vez más desconocidos y peor usados.

7.3. Objeto Session.

Este objeto es del tipo integrado, es decir no tenemos que crear una instancia del mismo para poder utilizarlo. Nos sirve para compartir información entre diferentes páginas, pero sin necesidad de compartirla entre diferentes clientes como sucede con el objeto *Application*, es decir, el objeto Session representa la sesión del usuario actual que esta utilizando la aplicación.

Como ya vimos anteriormente este objeto al igual que el *Application* nos permite crear variables globales cuyo valor podemos compartir en todas las paginas de nuestra aplicación, sin embargo, este valor no se comparte entre los clientes de la aplicación, ya que tienen el ámbito de sesión y solo pueden ser accedidas por dicho cliente; se usan para almacenar valores personalizados, como por ejemplo el nombre del usuario, opciones que ha utilizado, idioma, etc. y como estos valores son por cliente, no podemos almacenarlos en una variable de aplicación, es decir, cada usuario que ingrese a la aplicación tendrá un valor distintos para cada variable con respecto a otros usuarios..

El servidor crea automáticamente el objeto session cuando un usuario solicita una pagina Web de nuestra aplicación. Así mismo el servidor destruye la sesión cuando desde una página ASP se invoca el método *Abandon* o si el usuario tiene más de 20 minutos de inactividad en la aplicación. La sintaxis para la declaración de este objeto es la siguiente:

```
<%Session("Nom_variable") = valor%>
```

Veamos un ejemplo de cómo declarar una variable de sesión y como utilizarla con código HTML:

```
<%Session("nombre_usuario") = "Edgar Guerrero"%>
<font face= "arial" color="black" size="2">Hola: <%=session("nombre_usuario ")%></font>
```

En este caso cada que un usuario nuevo entre vera su nombre escrito en el mensaje de bienvenida de este ejemplo, ya sea que el nombre lo introdujo el propio usuario o fue obtenido de base de datos.

El contenido del objeto Session de cada visitante se guarda en la memoria del servidor, por lo que éste la recupera eliminando las sesiones de los visitantes que han abandonado el sitio. Esto se consigue de dos formas: la primera, automática, cuando caduca el tiempo de sesión (Session TimeOut) que tiene un valor por defecto (20 minutos) y que representa el tiempo que ha estado el usuario sin hacer ninguna; la segunda, llamando desde una página ASP al método Abandon del objeto Session, así:

```
<% Session.Abandon %>
```

Cuando se ejecuta este método, el servidor Web vacía el objeto y borra el ID de sesión del usuario que ha ejecutado ese código. El tiempo de sesión puede cambiarse, bien para todo el sitio Web desde el Administrador de IIS, o bien para una única sesión, especificándolo en una página ASP así:

```
<% Session.TimeOut = 5 %>
```

7.4. Objeto Request.

Este objeto es del tipo integrado, es decir, no tenemos que crear una instancia del mismo para poder utilizarlo. Este objeto sirve para obtener la información que puede pasársele a las páginas ASP.

El objeto *Request* nos devuelve información del usuario que ha sido enviada por medio de formularios, por parámetro o a partir de cookies. Dependiendo de la forma en que enviemos los datos al servidor tendremos que utilizar alguna colección de este objeto para recoger dicha información. Las más típicas son:

1. *QUERYSTRING.*
2. *FORM.*
3. *COOKIES.*

Colección QueryString.

La colección *QueryString* como ya vimos en el capitulo anterior, recupera la información

enviada al servidor mediante formulario utilizando el método *Get* o pasando la información de forma manual después de la dirección URL. La sintaxis del objeto utilizando esta colección es la siguiente:

Request.QueryString(“nom_elemento”)

Vemos un ejemplo del uso de esta colección:

```
<%
        If Request.QueryString("envio_datos") = "SI" Then
                Dim nombre, apellido_p, apellido_m, edad, sexo, estado
                nombre = Request.QueryString("txtNombre")
                apellido_p = Request.QueryString("txtApePat")
                apellido_m = Request.QueryString("txtApeMat")
                edad = Request.QueryString("txtEdad")
                If Request.QueryString("rbtSexo") = "F" Then
                        sexo = "Mujer"
                ElseIf Request.QueryString("rbtSexo") = "M" Then
                        sexo = "Hombre"
                Else
                        sexo = "No Definido"
                End If
                estado = Request.QueryString("cmbEstado")
        End If
%>
```

[Vease Anexo A18]

En el ejemplo del formulario anterior se utiliza el método *Get* para enviar la información (method="get"), y se envía a la misma pagina, es decir a si misma, al escribir el usuario sus datos se envía la siguiente petición al servidor:

http://localhost/objeto_request_querystring.asp?txtNombre=Edgar+Antonio&txtApePat=Guerrero&txtApeMat=Gonzalez&txtEdad=26&rbtSexo=M&cmbEstado=Hidalgo&btnAceptar=Aceptar&envio_datos=SI

Esta colección tiene la propiedad *Count*, que sirve para contar el número de veces que aparece un determinado tipo de valor. Por ejemplo podemos utilizar esta colección para contar los valores siguientes:
http://localhost/objeto_request_querystring.asp?comida=Mexicana&comida=China&comida=Tailandesa&comida=Peruana&comida=Italiana

Para obtener los valores anteriores, podemos utilizar el siguiente código:

```
<%
total = Request.QueryString(“comida”).count
For i = 1 To total
  Response.Write “Tipo de Comida: “ & Request.QueryString(“comida”).Count&”<BR>”
Next
%>
```

Lo anterior mostraría la siguiente secuencia:

Tipo de Comida: Mexicana
Tipo de Comida: China
Tipo de Comida: Japonesa

Colección Form.

Utilizamos la colección *Form* para acceder a la información que es enviada en un formulario utilizando el método *Post*. Con este método no se añaden los elementos del formulario a la cadena de consulta, sino que forman parte del cuerpo de la petición *http*.

El uso de esta colección es similar a *QueryString*, ya que implementa la misma interfaz. Este método es el que generalmente se utiliza en el envío de formularios grandes y complejos, ya que permite enviar un número casi ilimitado de caracteres a un servidor. La sintaxis del objeto utilizando la colección *Form* es la siguiente:

Request.Form("nom_elemento")

En el siguiente ejemplo utilizaremos el mismo formulario que usamos en la colección *QueryString*, pero utilizando el método *post* para enviar la información:

```
<%
        If Request.Form("envio_datos") = "SI" Then
                Dim nombre, apellido_p, apellido_m, edad, sexo, estado
                nombre = Request.Form("txtNombre")
                apellido_p = Request.Form("txtApePat")
                apellido_m = Request.Form("txtApeMat")
                edad = Request.Form("txtEdad")
                If Request.Form("rbtSexo") = "F" Then
                        sexo = "Mujer"
                ElseIf Request.Form("rbtSexo") = "M" Then
                        sexo = "Hombre"
                Else
                        sexo = "No Definido"
                End If
                estado = Request.Form("cmbEstado")
        End If
%>
```

[Vease Anexo A19]

En este ejemplo podemos ver que es el mismo formulario de la colección *QueryString*, sin embargo utiliza otro método para enviar la información: *Post* (method="post"), sin embargo los datos enviados ya no se ven en la URL, sino que van dentro del cuerpo del formulario y la forma de acceder a los datos también es diferente.

Colección Cookies.

Las cookies son el mecanismo que nos permite guardar información relativa a un usuario a lo largo de sus distintos accesos a nuestras páginas. Estos elementos surgieron por las características del protocolo *http*, ya que cuando un cliente visita un sitio Web se crean conexiones y desconexiones en el transito de información en la red.

Las colecciones anteriores, *QueryString* y *Form*, solo pueden acceder una vez a la información enviada por el cliente, justo cuando el servidor recibe la petición. Existen muchas situaciones donde es útil mantener la información persistente en todas las peticiones que realice un cliente. La colección *cookies* es una solución a esta necesidad, ya que se almacenan en los equipos de los clientes, utilizándolo como almacén de información. La *cookie* se almacena dentro de un fichero en el ordenador del cliente, con lo cual el servidor tiene acceso a ella en diferentes peticiones.

Esto hay que tenerlo en cuenta ya que el ordenador del cliente se utiliza para grabar estos datos, no se almacenan en el servidor, lo cual trae como contrapartida las posibles faltas de integridad de datos al no poder controlarlos en la computadora del cliente.

Las *cookies* se transmiten en las cabeceras cuando se realiza la comunicación http y es el explorador el encargado de almacenarlas. Las cookies se implementan como una colección y se usan de manera similar a las colecciones anteriormente descritas, en concreto esta colección es de solo lectura. La sintaxis para poder utilizar el valor de una *cookie* es la siguiente:

```
Request.Cookies("nombre_cookie")
```

Las *cookies* las podemos recuperar en formato de colección ***cookies***, sin embargo los elementos de dicha colección no forman estructuras colección, sino ***Diccionario***. Para ello hacemos uso del atributo ***HasKeys***, el cual nos indica si una cookie posee claves, la sintaxis es la siguiente:

```
Request.Cookies("nombre_cookie").HasKeys
```

Si al recuperar la cookie posee un diccionario de claves, el valor anterior se evalúa a ***true***, en caso contrario devuelve ***false***. Un ejemplo de cómo usar una *cookie* que contenga un diccionario de claves es el siguiente:

```
Request.Cookies("nombre")("apellidoP")
Request.Cookies("nombre")("apellidoM")
```

En el siguiente ejemplo veremos un ejemplo sencillo del uso de las cookies con el objeto Request, almacenaremos el nombre de todos los usuarios que accedan a nuestra página para que en las próximas ocasiones que entren les aparezca un mensaje saludándolos con su nombre (vease figuras 7.2 y 7.3).

```
<%
If Request.Form.Count = 0 Then
  If Request.Cookies("registro").HasKeys = 0 Then%>
            <html>
               ....
            </html>
<%
  Else
            Dim iCont
            iCont = Request.Cookies("registro")("entradas")
            Response.Cookies("registro")("entradas") = iCont + 1
            Response.Write "Bienvenido: " & Request.Cookies("registro")("nombre") &"<br>"
            Response.Write "Has entrado " & iCont & " veces"
  End If
Else
  Response.Cookies("registro")("nombre") = Request.Form("nombre")
  Response.Cookies("registro")("entradas") = 1
  Response.Write "Gracias por Registrarse, "
  Response.Write Request.Cookies("registro")("nombre")&"<br><br>"%>
<%End If%>
```

[Vease Anexo A20]

Una vez que el cliente se registre en el sitio, el resto de las veces que entre a la página,

recibirá una saludo por el nombre que indico, diciéndole las veces que ha entrado (vease figura 7.4).

Figura 7.2 Introducir el valor que va a tener la *cookie*.

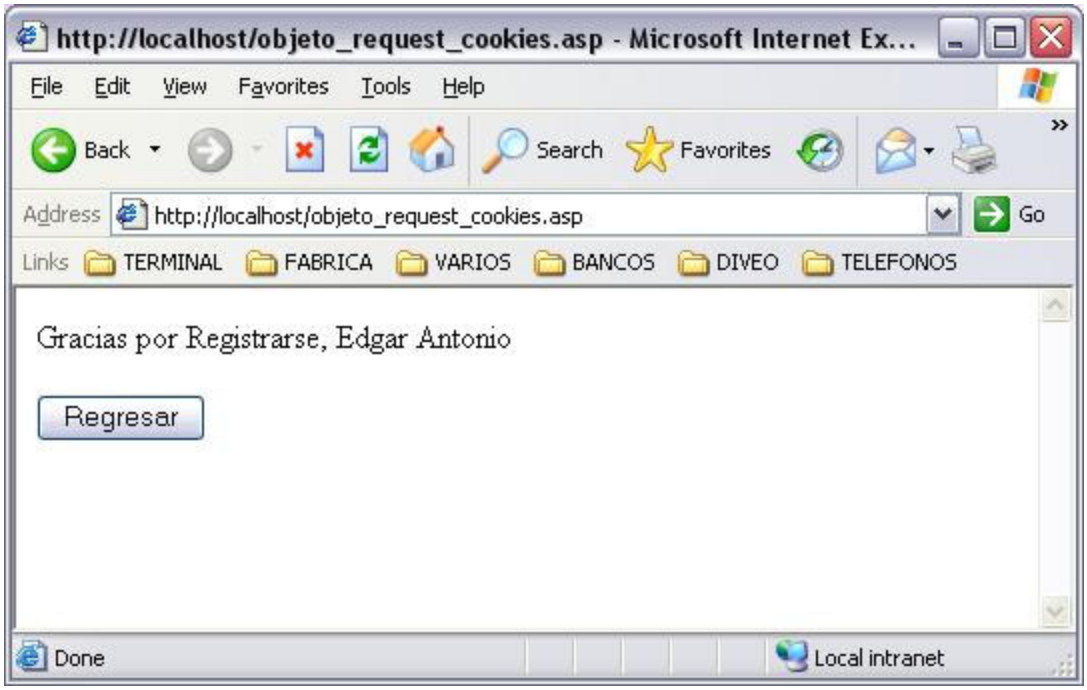


Figura 7.3 La *cookie* se crea con el valor que se introdujo.

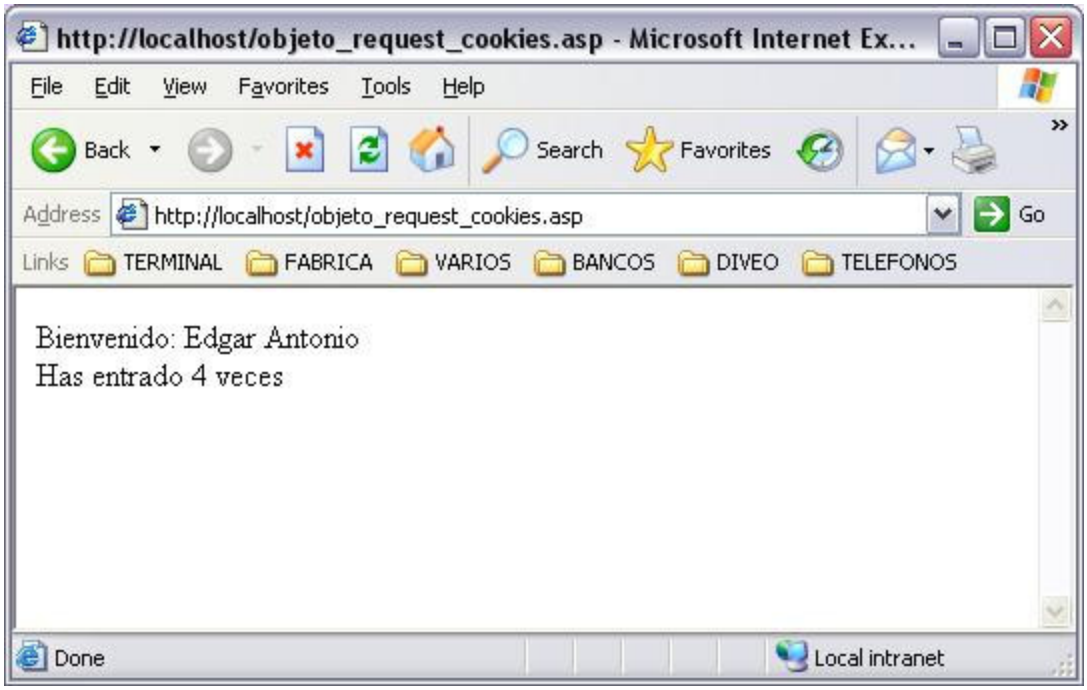


Figura 7.4 Cada que el cliente vuelve a entrar a la página ya no es necesario volver a introducir el valor, queda registrado en la *cookie* y también se almacena el número de veces que entra.

7.5. Objeto Response.

Ya vimos en el punto anterior que el objeto Request gestiona todo lo relativo a entrada de datos por parte del usuario, provenientes de otra URL, de un formulario, del propio servidor o del explorador.

Vamos ahora a explicar cómo después de procesar los datos, podemos imprimir estos en pantalla, inscribirlos en las cookies o enviar al usuario a una u otra pagina. Es decir, queda por definir la forma en la que ASP regula el contenido que es enviado al navegador.

Esta función es realizada por el objeto *Response* el cual ha sido utilizado en algunos ejemplos de capítulos anteriores, concretamente utilizándolo con el método *Write*, el cual vamos a explicar mas adelante.

Este objeto ASP es del tipo integrado, es decir, no tenemos que crear una instancia del mismo para poder utilizarlo. Este Objeto se usa para representar los datos que el servidor envía al explorador del cliente, es decir, representa la respuesta (información de salida) que el servidor Web da al explorador, para redireccionar a otros recursos, establecer valores de cookies, etc.

La sintaxis general para utilizar este objeto es la siguiente:

Response.metodo("cadena")

El objeto Response es uno de los más importantes puesto que es el que se encarga de presentar la información que producen los ASP en el documento que se va a mandar al cliente.

Los métodos principales del objeto Response son los siguientes:

Método Write.

Este método lo hemos estado utilizando en ejemplos anteriores, ya que sirve para escribir en el explorador la cadena especificada en la salida. También se pueden enviar etiquetas HTML y concatenar variables con el operador & (VBScript). Ejemplo:

```
<%Response.Write ("<b>hola!!!</b>")%>
```

Una cadena es cualquier combinación de caracteres ASCII, poniéndolos entre *comillas dobles*. Este método es tan comúnmente utilizado que se puede usar este método de manera implícita: *<% = variable %> es lo mismo que <%response.write variable%>,* es decir el símbolo = sustituye a *Response.Write().*

Método Redirect.

El método *Redirect* hace que el explorador intente conectar con una dirección URL distinta y envía al usuario automáticamente a la dirección que nosotros decidamos. Este método es útil en muchas aplicaciones en las cuales es necesario dirigir al usuario hacia otra dirección después de determinado tratamiento de los datos o simplemente como método de actualización de enlaces antiguos. La sintaxis para la utilización de este método es la siguiente:

```
<%Response.Redirect ("http://www.tralcom.com")%>
                    o
<%Response.Redirect ("tu_pagina.asp")%>
```

Esta función, puede ser empleada en scripts en los que enviamos al visitante de nuestro sitio a una u otra página en función del navegador que utiliza o la lengua que prefiere por ejemplo. También es muy útil para realizar páginas "escondidas" que realizan una determinada función sin mostrar ni texto ni imágenes y nos envían a continuación a otra página que es en realidad la que nosotros recibimos en el navegador.

Método End.

Detiene el procesamiento de la pagina ASP y devuelve el resultado acumulado hasta ese momento al explorador del cliente. La sintaxis para utilizarlo es la siguiente:

```
<%Response.End%>
```

Colección Cookies.

Ya vimos en el punto anterior la colección cookies, la cual son un mecanismo que nos permite guardar información relativa a un usuario a lo largo de sus distintos accesos a nuestras páginas en forma de archivo de texto. En este objeto también se utiliza esta colección y sirve para crear y modificar cookies en el explorador del cliente y su sintaxis es la siguiente:

- Ejemplo de una cookie simple:

```
<% Response.Cookies(“color”)=”morado” %>
```

- Ejemplo de una cookie con claves:

```
<% Response.Cookies("color")("fondo")="morado"%>
<% Response.Cookies("color")("texto")="blanco"%>
```

Cuando usamos el objeto *Response* para escribir una cookie, si esta ya existía se sobrescribe el valor de esta. El objeto *Request* tiene acceso a las cookies en modo lectura a su contenido y el objeto *Response* si puede alterar dicho contenido e incluso eliminar la cookie.

7.6. Objeto Server.

El objeto Server es de los más importantes para el manejo de sitios Web en ASP. Gracias a este podemos consultar aspectos del servidor y trabajar directamente con sus características. Este objeto solo posee propiedades y métodos, no tiene colecciones, veamos las más comúnmente utilizadas:

- *Propiedad ScriptTimeout*
- *Método CreateObject*
- *Método Execute*
- *Método HTMLEncode*
- *Método Transfer*
- *Método URLEncode*

Propiedad ScriptTimeout.

Esta propiedad se utiliza para determinar el tiempo máximo que se permite para la ejecución de una pagina ASP, el valor por defecto es de 90 segundos. Esta propiedad también se puede especificar en la administración del IIS. El valor que se especifique desde el código en ASP no puede ser menor al indicado en la administración del IIS.

Al programar con ASP construimos código HTML de forma dinámica, lo cual puede provocar que cometamos errores. Un error común es la ejecución de un bucle infinito en una pagina ASP, el sistema de seguridad que implementa IIS para evitar perder el control sobre una ejecución de este tipo es el tiempo de espera, gracias a esto, aunque codifiquemos erróneamente una pagina, cuando expire el tiempo, el servidor Web elimina el proceso, destruyendo así ese hilo de ejecución. La sintaxis para la utilización de esta colección es la siguiente:

```
<%Server.ScriptTimeout = No_Segundos%>
```

Esta propiedad es de lectura y escritura, la figura 7.5 muestra el tiempo de ejecución de la página, lo que cual se puede realizar de la siguiente forma:

```
<%
Server.SciptTimeout = 120
Response.Write "El tiempo de ejecución de la pagina es: " & Server.ScriptTimeout & " segundos."
%>
```

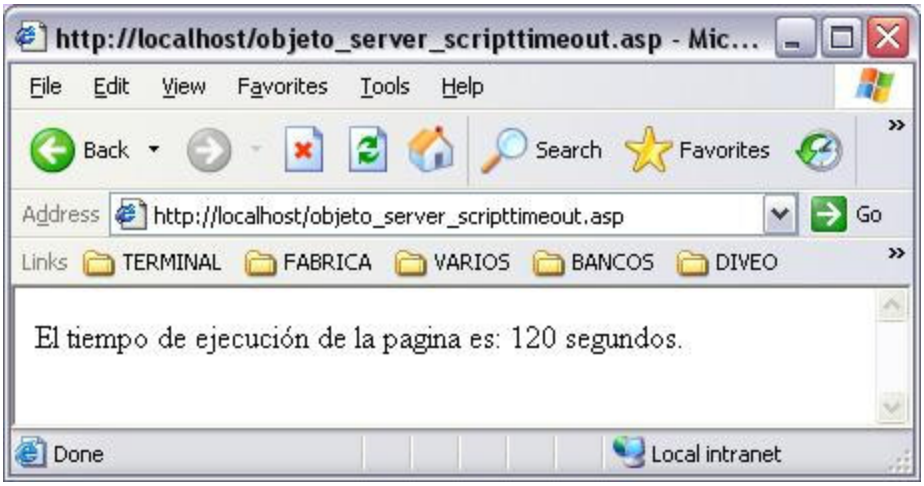


Figura 7.5 Ejemplo del Objeto *Server* utilizando la propiedad *ScriptTimeout*.

Método CreateObject.

El lenguaje ASP esta abierto a componentes, procedentes de la instalación del IIS, desarrollados por terceros o por el desarrollador, estos objetos son denominados *componentes de servidor*. Gracias a la arquitectura de objetos COM desarrollada por Microsoft se puede enriquecer la programación en ASP con nuevas funcionalidades encapsuladas en objetos.

El estándar COM tiene una serie de reglas para que dichos componentes puedan utilizarse desde ASP. Debemos crear una instancia del objeto que vayamos a utilizar, para esto podemos utilizar el método *CreateObject*.

Este es uno de los métodos más importantes y nos permite crear objetos a partir de los cuales vamos a trabajar. Como único parámetro se especifica el nombre del objeto a crear. Al crear el objeto, éste se le asigna a la variable establecida por medio de la palabra reservada *SET*, que hereda todos los métodos, propiedades y eventos del objeto creado.

La sintaxis para utilizar este método es la siguiente:

<%Server.CreateObject(ProgID)%>

Donde *ProgID* especifica el tipo de objeto que vamos a crear para poder utilizarlo. *ProgID* es una cadena de texto donde indicamos los parámetros que nos proporciona el fabricante del componente. Los objetos creados con este método tienen alcance de página, esto significa que al finalizar la ejecución de la página ASP dichos objetos se destruirán automáticamente.

Veamos un ejemplo de la utilización de este método para el acceso a datos, donde el

ProgID utilizado es *ADODB.Connection*:

```
<%
'Instanciamos el objeto ADODB.Connection
Set objConexion = Server.CreateObject("ADODB.Connection")
'Ya que instanciamos podemos usar sus métodos y propiedades
objConexion.Open = "Mi_BaseDatos"
..
..
'Si ya no usamos el objetos, lo destruimos
Set objConexion = Nothing
%>
```

Método Execute.

Este método llama a un archivo ASP y lo ejecuta como si fuera parte del archivo donde se realizo la llamada, es decir, ejecuta una pagina ASP dentro de otra. El parámetro que necesita es la ruta de la página que va a ejecutar. La sintaxis para poder utilizar este método es la siguiente:

<%Server.Execute(Ruta)%>

En Ruta especificamos la ubicación del archivo ASP que queremos ejecutar. Este método permite modularizar nuestra aplicación en elementos que realizan determinadas tareas, ya que podemos llamar a esos módulos cuando sea necesario ejecutar algún código en específico.

El siguiente ejemplo muestra la utilización de este método. Tenemos la página objeto_server_execute.asp y la página que ya vimos en un ejemplo anterior objeto_request_form.asp, lo único que hace la primera página es llamar todo el contenido de la segunda mediante el método *Execute*, lo cual parecería que esta definido en la misma página(vease figura 7.6).

```
<%
Response.Write "Esta pagina utiliza el método Execute, llamando la pagina objeto_request_form.asp"
Server.Execute "objeto_request_form.asp"
%>
```

La utilización de este método es una alternativa a la utilización de las librerías que vimos en el capitulo 5.

Método HTMLEncode.

Este método nos sirve para evitar que la cadena enviada como parámetro sea interpretada como lenguaje HTML. La sintaxis de este método es la siguiente:

<%Server.HTMLEncode(cadena)%>

Veamos un ejemplo imprimiendo una cadena en pantalla utilizando código HTML con y sin usar este método(vease figura 7.7).

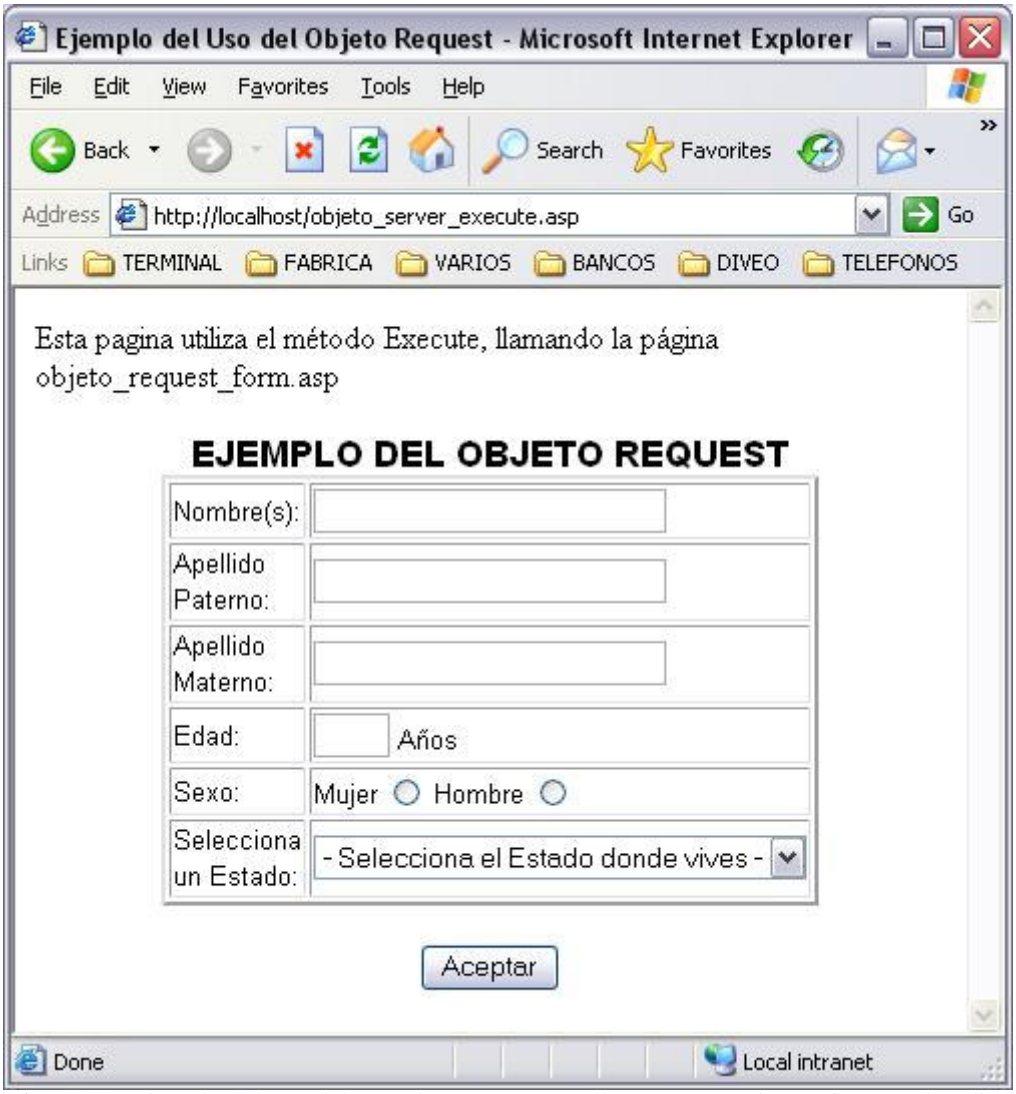


Figura 7.6 Ejemplo del uso del método Execute para llamar otra página y ejecutar el código existente en ella.

```
<%
Response.write ("<b><u>Edgar Guerrero</u></b><br><br>")
Response.write Server.HTMLEncode("<b><u>Edgar Guerrero</u></b><br><br>")
%>
```

En este ejemplo podemos ver que cuando no aplicamos el método *HTMLEncode*, interpreta el código HTML y formatea la cadena en el explorador, sin embargo al aplicarlo ignora dicho código y lo imprime en pantalla tal cual esta escrito en el programa.

Método Transfer.

Este método sirve para enviar a otra página toda la información obtenida al procesar la página de donde es enviada. La sintaxis de este método es la siguiente:

<%Server.Transfer(Ruta)%>

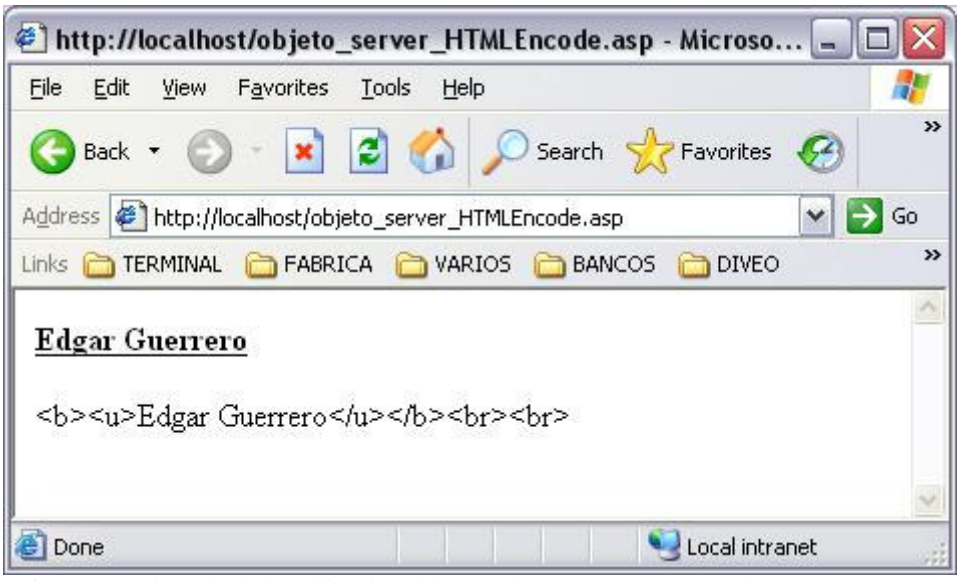


Figura 7.7 Ejemplo de la utilización del método *Server.HTMLEncode*.

Ruta es la localización de la pagina ASP a la cual se va a transferir el control de la ejecución. Al realizar esta transferencia se incluye toda la información de estado de todos los objetos integrados, variables y los valores almacenados en las colecciones del objeto *Request*.

Método URLEncode.

Este método se utiliza para aplicar las reglas de codificación URL, incluyendo los caracteres de escape, a la cadena especificada. La sintaxis de este método es la siguiente:

<%Server.URLEncode(cadena)%>

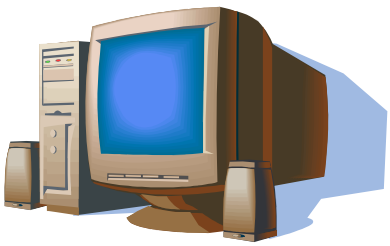
La utilización de este método es importante para pasar parámetros por QueryString que lleven espacios en blanco y otros caracteres que puedan ser codificados, por ejemplo:

location.href = "objeto_server_URLEncode.asp?nombre=<%=Server.URLEncode("Edgar Guerrero")%>"

Enviara la cadena especificada correctamente codificada para que ningún dato se pierda.

CAPITULO 8

El Archivo Global.asa.



En este capitulo se hablara del archivo de configuración global.asa y del uso que se le da dentro de una aplicación de ASP.

8.1. Introducción.

El archivo *Global.asa* es un fichero especial de nuestro sitio Web, es un fichero de texto que debe llamarse obligatoriamente así, *global.asa* y estar situado en el directorio raíz de nuestro servidor Web, es decir, en el directorio de inicio de nuestras páginas, dicho archivo es importante para toda aplicación Web, ya que es un lugar donde podemos inicializar objetos y variables globales a la aplicación o sesión (**asa – Active Server Application**).

Es un archivo de comandos que nos permite la automatización de los cuatro eventos básicos de nuestro servidor que podemos programar:

- *Application_OnStart*
- *Application_OnEnd*
- *Session_OnStart*
- *Session_OnEnd*

La utilidad de los eventos anteriores estriba en las posibilidades de poder crear objetos, inicializar variables, totalizar resultados que se han producido a lo largo de la sesión o aplicación, liberar recursos, etc.

Este archivo se programa también en Visual Basic Script(o en algún otro lenguaje de script que sea soportado en ASP), pero en lugar de utilizar las etiquetas <% y %, las cuales no se pueden usar aquí, debemos utilizar las etiquetas de <*script*> para que funcione correctamente el código declarado en dicho archivo

La sintaxis de declaración de este archivo es la siguiente:

```
<SCRIPT LANGUAGE="VBScript" RUNAT="Server">
' Declaración de los eventos del archivo Global.asa
Sub Application_OnStart
 Trozo de código
End Sub

Sub Application_OnEnd
 Trozo de código
End Sub

Sub Session_OnStart
 Trozo de código
End Sub

Sub Session_OnEnd
 Trozo de código
End Sub

</SCRIPT>
```

Al inicio del desarrollo de una aplicación de ASP se debe crear este archivo, incluso si no se piensa escribir ninguna secuencia de comandos en el. En este caso se puede dejar sin contenido los eventos mencionados anteriormente.

8.2. Evento Application_OnStart

Este evento se dispara cuando el primer usuario entra a nuestra aplicación Web y antes de crear su sesión., por lo tanto, se produce antes del evento *Session_OnStart*. Este procedimiento se utiliza para hacer las inicializaciones necesarias a nivel de aplicación, por ejemplo para el caso de un contador de visitas, entre otros.

La sintaxis para la utilización de este evento es la siguiente:

```
Sub Application_OnStart
'Trozo de código que queremos que se ejecute al inicio de la aplicación
End Sub
```

Volviendo a utilizar el ejemplo del contador de visitas que se vio en el capitulo anterior del objeto Application, utilizaremos este evento para inicializar la variable en el archivo global.asa:

```
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
Sub Application_OnStart
Application("n_visitas") = 0 'Inicialización de la variable para controlar el numero de visitantes al sitio.
End Sub
</SCRIPT>
```

8.3. Evento Application_OnEnd

Este evento se dispara cuando el último cliente sale de la aplicación y termina su sesión, por lo tanto se produce después del evento Session_OnEnd. Se puede programar por ejemplo para que guarde todas las variables de aplicación en una base de datos y así poder recuperar posteriormente los valores actuales.

Su sintaxis es la siguiente:

```
Sub Applicaton_OnEnd
'Trozo de código que queremos que se ejecute al terminar la aplicación
End Sub
```

8.4. Evento Session_OnStart

Este evento se dispara cuando un usuario solicita una página de la aplicación y se crea su sesión. Este evento se utiliza para poder inicializar variables y objetos globales igual que con el evento Application_OnStart, pero a nivel de sesión, es decir una para cada usuario que entre al sitio.

La sintaxis de este evento es la siguiente:

```
Sub Session_Onstart
'Trozo de código que queremos que se ejecute al inicio de la sesión.
End Sub
```

Veamos un ejemplo de la utilización de este evento:

```
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
Sub Session_OnStart
'Redirecciona a un usuario a otra pagina si por ejemplo se cambio la anterior.
```

```
Response.Redirect "nueva_pagina_inicio.asp"
End Sub
</SCRIPT>
```

8.5. Evento Session_OnEnd

Este evento se dispara cuando un usuario abandona su sesión o se supera el tiempo de espera de la misma. Para poder detectar este tiempo se hace lo siguiente: cada vez que un usuario realiza una petición, se inicia un contador de tiempo, el cual es reiniciado cada vez que se detecta cualquier actividad de ese usuario, si el contador alcanza el timeout de la sesión y el usuario no ha hecho ninguna actividad, se considera que dicho usuario ha abandonado la sesión.

El timeout de la sesión es de 20 minutos por defecto, sin embargo este es configurable. Por ejemplo, si quisiéramos aumentar el timeout de la sesión de nuestra aplicación a 30 minutos, escribimos lo siguiente:

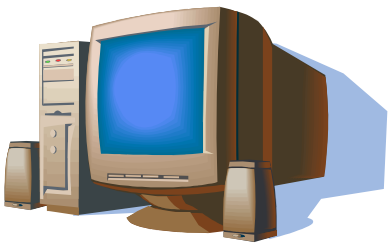
```
Session.Timeout = 30
```

La sintaxis del evento Session_OnEnd es la siguiente:

```
Sub Session_OnEnd
'Trozo de código que queremos que se ejecute al terminar la sesión.
End Sub
```

CAPITULO 9

Fuentes de Datos ODBC.



En este capitulo se hablara del Open DataBase Connectivity (ODBC), como administrar y crearlo, para que sirve y como utilizarlo en una aplicación ASP.

9.1. Introducción.

Si escribimos una aplicación para acceder a las tablas de una DB de Access por ejemplo, y después queremos que la misma aplicación, y sin reescribir nada, utilice tablas de SQL Server u otra DB cualquiera, no funcionaria nuestra aplicación, ya que estaría diseñada para un motor concreto, y no sabría dialogar con el otro. Pero si hubiera un elemento que por un lado sea siempre igual, y por el otro sea capaz de dialogar con una DB concreta, solo tendríamos que ir cambiando dicho elemento, y nuestra aplicación siempre funcionaría sin importar el motor de BD de que estemos utilizando. A esas piezas intercambiables las llamaremos *fuentes de datos de ODBC*. Por un lado el ODBC provee de unas características siempre homogéneas, y por el otro permite distintos controladores que aseguran la conectividad de la aplicación con diferentes bases de datos

ODBC es una utilidad general de Windows NT o superior que resulta necesaria para poder acceder de forma sencilla a diferentes bases de datos, por lo que su funcionalidad es importante en la programación de paginas ASP.

ODBC es una interfaz que permite a las aplicaciones acceder a distintos tipos de BD, para ello se necesitan los diferentes controladores ODBC con los que se obtiene información de los orígenes de datos. La Figura 9.1 muestra la ventana *ODCB Data Source Administrator*, la cual contiene la pestaña *Drivers*, en donde podemos ver los diferentes controladores que podemos utilizar para acceder a la BD.

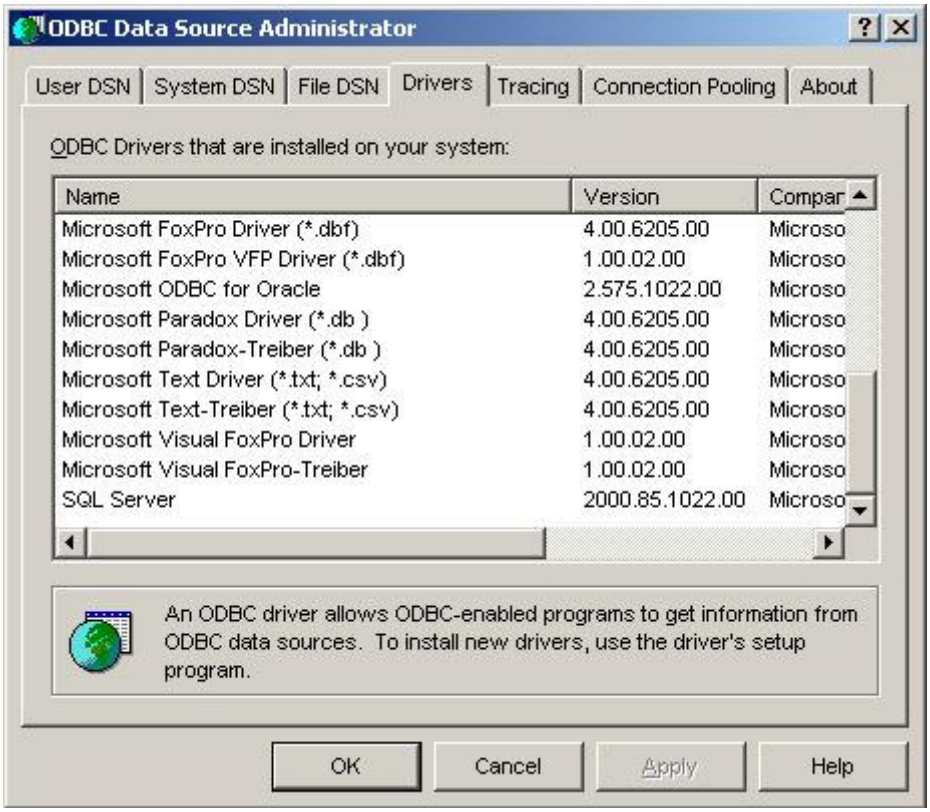


Figura 9.1 Controladores soportados en ODBC.

9.2. ¿Qué es ODBC?

ODBC son las siglas de **Open DataBase Connectivity(Conectividad Abierta de Bases de Datos)**, un estándar de acceso a Bases de Datos desarrollado por Microsoft, el objetivo de *ODBC* es que sea posible acceder a cualquier dato de cualquier aplicación, sin importar qué Base de Datos utilicemos para almacenar la información, *ODBC* logra esto al insertar una capa intermedia llamada manejador de Bases de Datos, entre la aplicación y la BD, el propósito de esta capa es traducir las consultas de datos de la aplicación en comandos que la BD entienda, es decir, es un intermediario entre bases de datos y aplicaciones, cuya tarea es sostener una conversación de preguntas y respuestas entre dos "sujetos" que no hablan el mismo idioma y que gestionan sus recursos de forma diferente.

En la figura 9.2 se muestra el esquema de cómo se comunica un ODBC a cualquier BD, especificándole el controlador que debe de utilizar para poder establecer la conexión:

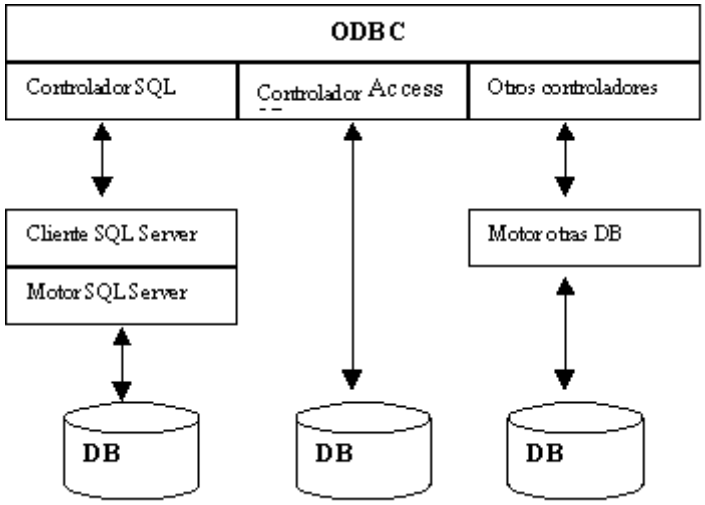


Figura 9.2 Esquema de conexión de un ODBC a BD.

9.3. Funcionamiento de ODBC

Para conectarse a la Base de Datos se crea un *DSN* dentro del *ODBC* que define los parámetros, ruta y características de la conexión según los datos que solicite el fabricante. *DSN* es un *Data Source Name (Nombre de Fuente de Datos)* que representa una fuente de datos configurada por el usuario para conectarse a una BD. Es decir, por cada conexión que se quiera establecer con la base de datos, es necesario especificar una serie de información que permitan al Controlador saber la manera de conectarse.

Existen tres tipos de *DSN* que podemos crear para conectarse al origen de datos:

1. *User DSN (Nombre del Origen de Datos de Usuario).* Este tipo de fuente de datos es utilizado para poder acceder al origen de datos exclusivamente por el usuario especificado.
2. *System DSN (Nombre del Origen de Datos de Sistema).* Con este tipo de fuente de datos, cualquier usuario que utilice la aplicación puede acceder a la BD. Este es el que se emplea mayoritariamente en las aplicaciones Web que implementan ASP y es el que se manejara como ejemplo en este capitulo.

3. *File DSN (Nombre del Origen de Datos de Archivo).* Con este tipo de fuente de datos, se crea un archivo con la extensión DSN y sirve así como origen de datos y puede ser distribuido a otros usuarios. Si crea este tipo de fuente de datos, normalmente se almacena en la ubicación predeterminada, que es c:\Archivos de programa\Archivos comunes\ODBC\Data Sources.

Un *DSN* permite en realidad definir la base de datos que será interrogada sin necesidad de pasar por la aplicación que hayamos utilizado para construirla, es decir, con simples llamadas y órdenes desde un programa podremos obtener los datos que buscamos sin necesidad de ejecutar el manejador de la base de, el cual, evidentemente, no tiene por qué encontrarse en el servidor donde estemos trabajando.

9.4. Crear un ODBC

Vamos a ver ahora como crearlo, es un sencillo proceso, lo crearemos para una BD de SQL Server:

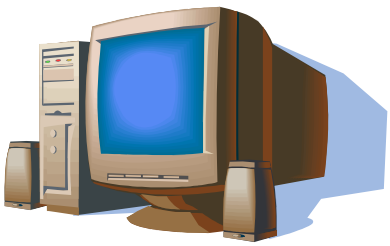
- Seleccionamos el icono *Administrative Tools (Herramientas Administrativas)* en el Panel de Control de Windows Y Seleccionamos el icono ODBC dentro de la carpeta de Herramientas Administrativas.
- En el cuadro de dialogo que se muestra, seleccionamos la pestaña System DSN, y hacemos click en Add (Agregar).
- A continuación, seleccionamos el controlador que se va a utilizar de la lista que se despliega y damos click en finish, en este caso, seleccionamos el controlador de SQL Server.
- Indicamos a continuación el nombre que le vamos a asignar, que es con el cual lo vamos a ejecutar desde nuestra aplicación, una breve descripción, si se desea y/o requiere y el servidor al cual se conectara y damos click en Next (siguiente).
- A continuación indicamos el usuario y password con el cual nos validaremos al acceder a la BD(debe estar creado en la base de datos previamente) y al terminar damos click en Next (siguiente).
- En las siguientes ventanas que aparezcan, damos click en Next hasta terminar con la creación de nuestro DSN y tendremos creado nuestro acceso a BD.

Ya con el ODBC creado, podremos acceder a la BD desde nuestras paginas ASP y manipular la información que necesitemos, en este caso todos los usuarios tendrán la posibilidad de acceder utilizando el origen de datos *User_DSN*.

El DSN permite en realidad definir la base de datos a la cual nos conectaremos para obtener información, sin necesidad de pasar por la aplicación que hayamos utilizado para construirla, es decir, con simples llamadas y ordenes desde un programa podremos obtener los datos que buscamos sin necesidad de ejecutar el manejador de base de datos.

CAPITULO 10

Base de Datos.



En este capitulo se hablara del uso de una Base de Datos en SQL Server, como crear la BD y sus tablas, así como la estructura de información que se manejara en ella, los tipos de operaciones que se pueden realizar sobre la misma y cómo accederla mediante una aplicación en ASP.

10.1. Introducción.

Una de las mayores ventajas que nos ofrecen las páginas ASP es la facilidad de trabajar con bases de datos. Esto lo consigue gracias a la utilización de ActiveX Data Objects (ADO), *ADO* son un conjunto de objetos que nos permiten acceder a la base de datos independientemente del motor de base de datos que usemos, estos ejemplos usan SQL Server pero funcionarían igual si el motor de Base de Datos fuese MS Access, Oracle, etc. Tan solo habría que cambiar el driver. Lo anterior hace que unas simples páginas Web se conviertan en aplicaciones completas de gestión de datos. Con las bases de datos se consigue un dinamismo impresionante, a parte de ahorrarnos mucho trabajo.

Por ejemplo, si tuviéramos un sitio Web de noticias, con lenguaje HTML tendríamos que hacer una página por cada una de las noticias. Todas las páginas tendrían un diseño similar, lo único que cambiaria seria el contenido de la noticia. Utilizando una base de datos, solo tendríamos que hacer una solo página que leyera de la BD la noticia que deseáramos mostrar, pasándole un parámetro con el código de la noticia, la página sabría buscar en la BD la noticia a mostrar.

A parte de ahorrarnos tiempo de desarrollo con una BD, tendremos la enorme facilidad de realizar cambios. Si en el ejemplo anterior tuviéramos que realizar un cambio sobre en el formato de las noticias, tendríamos que cambiar cada pagina de cada noticia y cambiar una a una el formato. Con BD, solo habría necesidad de cambiar una página, la que tiene acceso a la base.
Una BD es un conjunto de datos estructurados, organizados independientemente de su utilización y su implementación en maquina accesibles en tiempo real y compatibles con usuarios concurrentes con necesidad de información diferente.

En este trabajo nos enfocaremos a la base de datos de SQL Server de Microsoft para crear y conectarnos a base de datos. SQL Server es posiblemente el origen de datos más popular utilizado con ASP y utiliza el lenguaje SQL para el manejo de los datos. Parte de la razón de su popularidad entre los desarrolladores es su estrecha integración con el sistema operativo Windows y es recomendable para aplicaciones mayores y con una gran afluencia de clientes por su gran potencia. Otro factor a su favor es su interfaz grafica de usuario (GUI) muy intuitiva.

SQL es un lenguaje estructurado de consultas (Structured Query Language), es un estándar definido dedicado a realizar consultas (querys) a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Es un lenguaje de sintaxis simple y muy potente, mediante el se puede recorrer, modificar o borrar registros de las tablas que contienen nuestros datos.
SQL es un lenguaje fácil de aprender y una herramienta completa para gestionar datos. Las peticiones sobre los datos se expresan mediante sentencias, que deben escribirse de acuerdo a las reglas sintácticas y semánticas de este lenguaje.

En este capitulo no haremos una descripcion completa del uso del lenguaje SQL, sino que mostraremos las sentencias sencillas del lenguaje y como utilizarlas con ASP, por lo que es responsabilidad del usuario que tiene en sus manos este trabajo, aprender SQL para sacar un buen rendimiento a las paginas ASP.

10.2. Crear una Base de Datos

Una base de datos en un sistema relacional está compuesta por un conjunto de tablas, que corresponden a las relaciones del modelo relacional. La creación de la base de datos consiste en la creación de las tablas que la componen. En realidad, antes de poder proceder a la creación de las tablas, normalmente hay que crear la base de datos, lo que a menudo significa definir un espacio de nombres separado para cada conjunto de tablas.

En el ejemplo que mostraremos a continuación, crearemos una base de datos llamada BD_ejemplo y dentro de esta base de datos dos tablas denominadas Facturas y Proveedores respectivamente:

Desde el SQL Enterprise Manager, le damos New Database en el menú que se despliega al oprimir el botón derecho del mouse(vease figura 10.1).

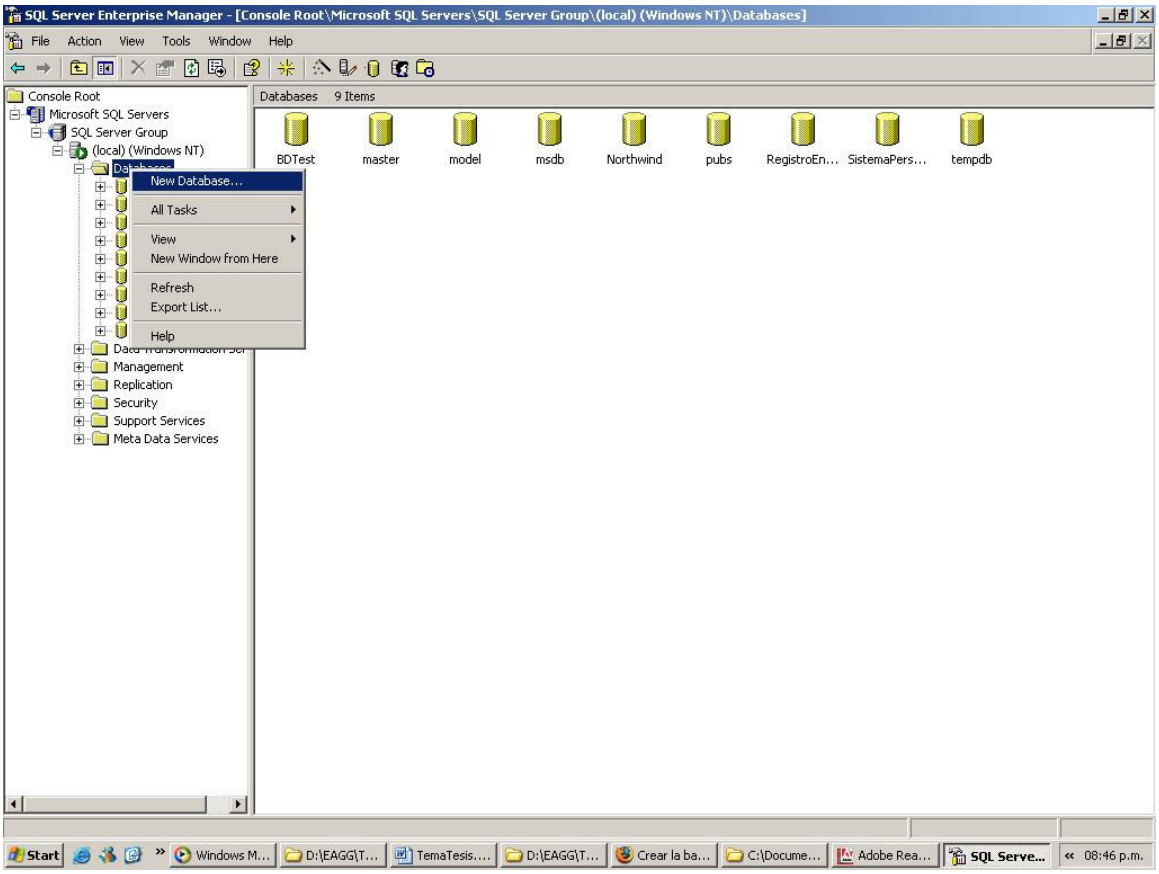


Figura 10.1 Creación de una Base de Datos.

En la ventana que se muestra, escribimos el nombre que le vamos a asignar a la base de datos que vamos a crear y pulsamos el botón OK y la base de datos la tendremos creada(vease figura 10.2).

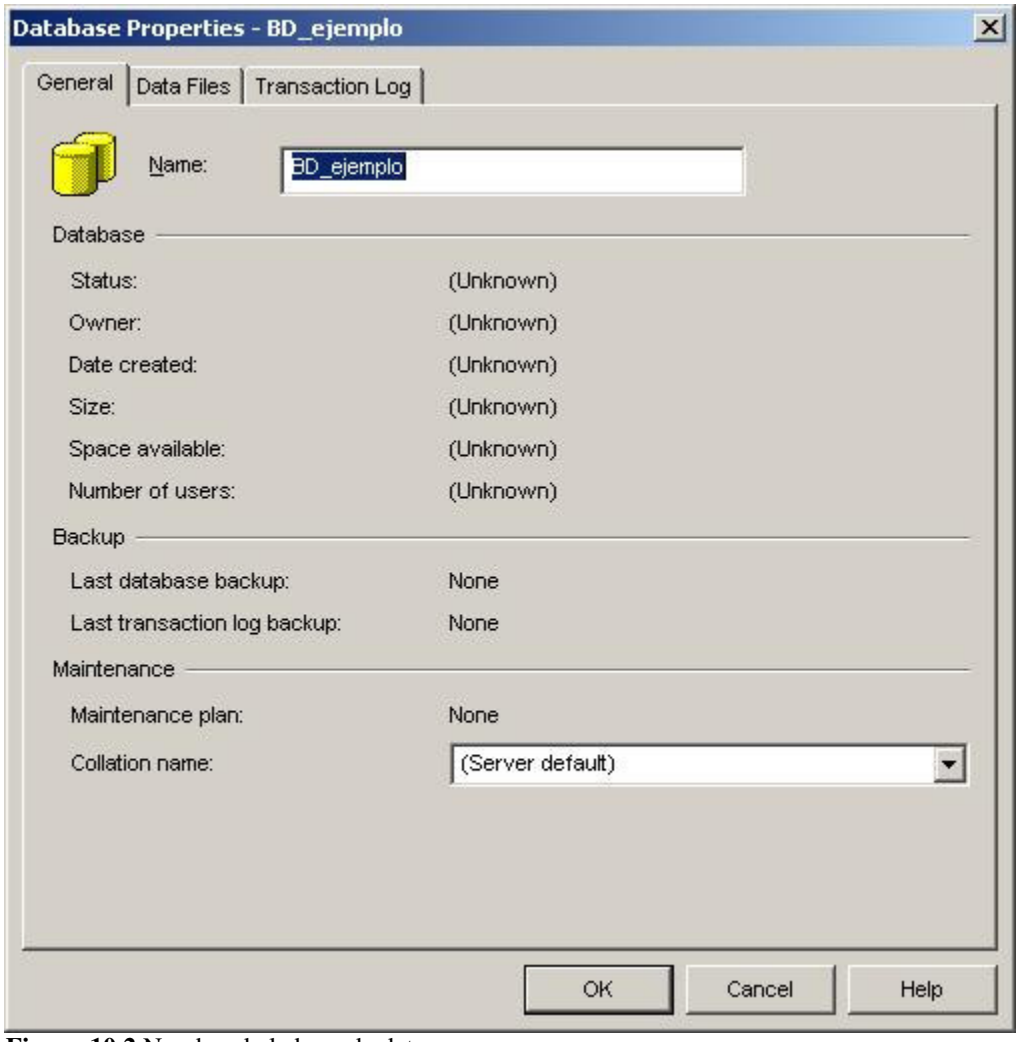


Figura 10.2 Nombre de la base de datos nueva.

También podemos crear nuestra base de datos utilizando la siguiente sintaxis:

CREATE DATABASE nombre_base_datos

Una vez creada la base de datos, lo siguiente es crear las tablas que va a contener y a almacenar la información, la creación de tablas mencionadas anteriormente (Facturas y Proveedores), se realiza de la siguiente manera:

Facturas:

En este ejemplo crearemos una tabla llamada Facturas que contiene 7 campos, y el campo cve_factura nos servirá para identificar inequívocamente una fila con el valor de dicho campo.

```
CREATE TABLE [dbo].[Facturas] (
  [cve_factura] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
  [descripcion] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
  [cve_proveedor] [int] NOT NULL ,
  [precio_factura] [money] NOT NULL ,
```

```
  [activa] [char] (2) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
  [fecha_alta] [datetime] NOT NULL ,
  [fecha_cancelacion] [datetime] NULL
) ON [PRIMARY]
```

Proveedores:

En este ejemplo crearemos una tabla llamada Proveedores que contiene 11 campos, y el campo cve_proveedor nos servirá para identificar inequívocamente una fila con el valor de dicho campo.

```
CREATE TABLE [dbo].[Proveedores] (
[cve_proveedor] [int] NOT NULL ,
[nombre_proveedor] [varchar](100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
[direccion] [varchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
[pais] [int] NOT NULL ,
[estado] [int] NOT NULL ,
[ciudad] [int] NOT NULL ,
[telefono] [int] NULL ,
[activo] [char] (2) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
[contacto] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
[fecha_registro] [datetime] NOT NULL ,
[fecha_baja] [datetime] NULL
) ON [PRIMARY]
```

Una vez creada la base de datos, podemos empezar a almacenar la información que necesitemos en ella, y manipularla de acuerdo a nuestras necesidades desde nuestras paginas ASP.

10.3. Seguridad.

La seguridad es una de las principales preocupaciones en la administración de sistemas y bases de datos, por la posibilidad de ataques externos. Lo primero que se debe de hacer es un plan de seguridad, en donde se identifique que usuarios pueden accesar a determinada información y que actividades pueden realizar en la base de datos. SQL Server se instala en Windows como un servicio y por tanto es necesario asignar un usuario para que inicie este servicio y para que trabaje en ese contexto de seguridad.

Modelo de Seguridad de SQL Server.

Para poder accesar a la información de una base de datos, un usuario tiene que pasar por dos niveles de autenticación: a nivel de SQL Server y a nivel de base de datos. Estos dos niveles son implementados utilizando nombres de usuario y cuentas, respectivamente. Un login valido es necesario para conectarse al SQL Server y una cuenta de usuario valida es necesaria para acceder a una base de datos.

- ***Login.*** Un nombre de login valido es necesario para conectarnos a SQL Server. Un login puede ser: Un login de Windows NT/2000/2003 al cual se le ha permitido el acceso a SQL Server. Un login de SQL Server que es manejado dentro del mismo SQL Server. Estos nombres de login son manejados dentro de una base de datos master. Esta es una de las razones del porque es esencial realizar una copia de seguridad después de adicionar nuevos logias a SQL Server.

- ***User.*** Una cuenta de usuario valida dentro de una base de datos es necesaria para acceder a dicha BD. Las cuentas de usuario son específicas a una base de datos. Todos los permisos y los propietarios de los objetos son controlados por la cuenta de usuario. Las logias de SQL Server son asociados a estas cuentas de usuario. Un login puede tener usuarios asociados en diferentes bases de datos, pero únicamente un usuario por base de datos.

Durante una solicitud de conexión, SQL Server verifica el nombre de login suministrado, para asegurar que el in esta autorizado para acceder a SQL Server. Este proceso de verificación es llamado autenticación. Las dos formas de autenticación proporcionadas por SQL Server son:

- ***Modo de Autenticación de Windows.*** Con este tipo de autenticación no es necesario especificar un nombre y contraseña de login para conectarnos a SQL Server, debido a que el acceso es controlado por nuestra cuenta de usuario de Windows(o el grupo al cual nuestra cuenta pertenece). Es necesario especificarle a SQL Server todas las cuentas o grupos de Windows que podrán conectarse.

- ***Modo de Autenticación Mixto.*** Este modo de autenticación permite a los usuarios conectarse, usando autentificación de Windows o de SQL Server. Primero tenemos que crear una cuenta y contraseña de login valida en SQL Server. Estas cuentas no están relacionadas a las de Windows. Con este modo de autenticación será necesario proporcionar el login y contraseña cada que nos conectemos a SQL Server. Si no se especifica un login y contraseña de SQL Server o solicitamos autenticación de Windows, nos autenticaremos utilizando la cuenta de Windows. Este es el modo de autenticación que utilizaremos en este trabajo. El modo de autenticación puede ser cambiado a través del Enterprise Manager(Administrador Corporativo), dando click derecho sobre el nombre del servidor y click en Propiedades, luego ir a la pestaña de seguridad y ahí se encuentra la opción.

Ahora veamos como crear un login y contraseña para poder accesar a SQL Server, accesamos a la carpeta de seguridad del servidor en el Enterprise Manager, a continuación damos click en *New Login* dando click derecho sobre *Logins*(vease figura 10.3).

En la ventana que se abre a continuación, escribimos el nombre del login que vamos a crear, la autenticación indicamos que es mediante SQL Server e indicamos la contraseña del login.

Seleccionamos la pestaña de Database Access(Acceso a base de datos) y le indicamos a que base(s) de datos queremos asociar el login, e indicamos los roles que tendrá la cuenta de acceso a la base de datos(ver figura 10.4).

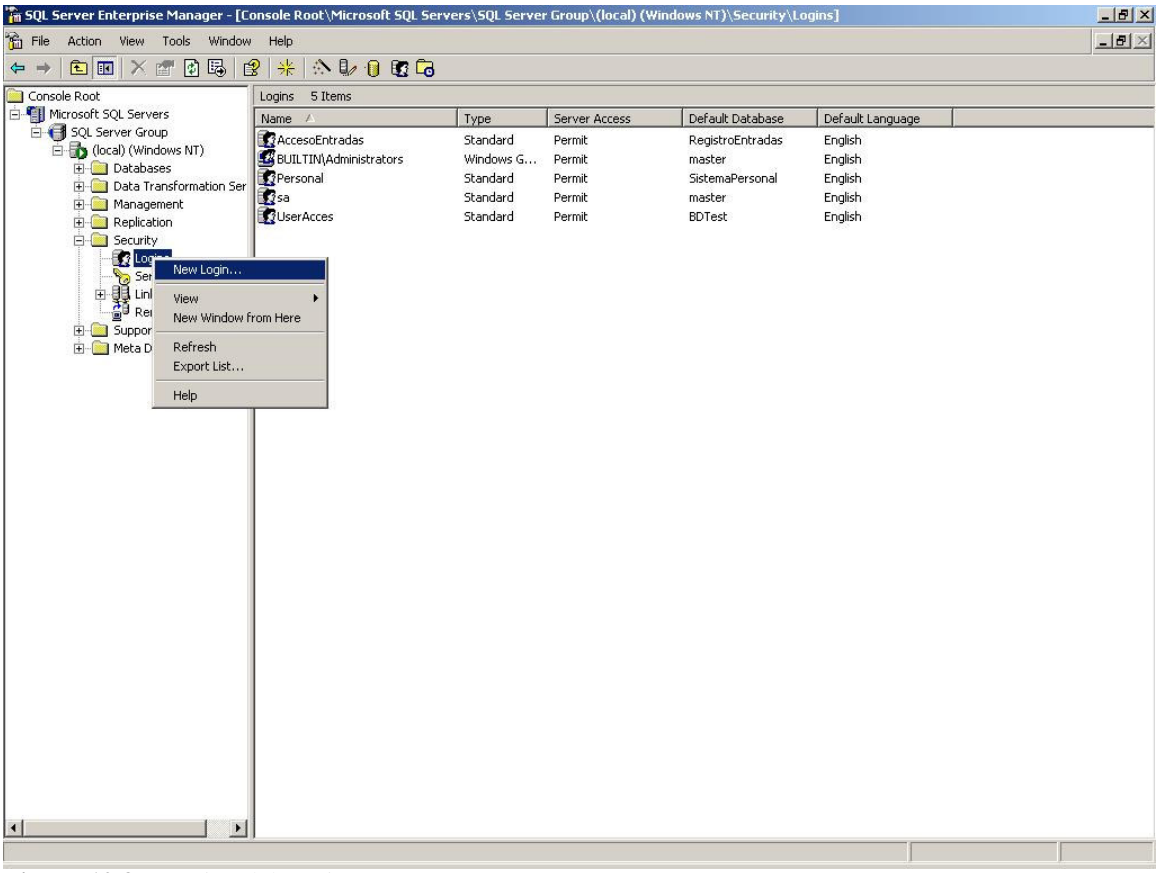


Figura 10.3 Creación del Login para accesar a SQL Server.

A continuación damos click en OK y nos pedirá confirmar la contraseña, una ves hecho esto, el login que acabamos de crear aparecerá en nuestro servidor y este será el que utilicemos al crear nuestro DSN(descrito en el capitulo anterior) para conectarnos a la base de datos:

10.4. Conexión a la Base de Datos

Ya que tenemos creada la base de datos en nuestro servidor, y el usuario con el cual vamos a accesar, el siguiente paso es conectarnos a la misma desde una página ASP.

Como ya vimos, para poder accesar a la BD necesitamos de los objetos *ADOB*, los cuales nos ofrecen una serie de métodos y propiedades para poder accesar fácilmente a la BD. Existen siete objetos para realizar dicho acceso, de estos siete objetos, podemos resaltar tres principalmente: *Connection*, *RecordSet* y *Command*. Estos tres objetos los podemos instanciar directamente desde nuestras páginas. El resto de los objetos son: *Field*, *Parameter*, *Property* y *Error*, los cuales utilizaremos como complementarios de los principales. Por ejemplo, obtendremos un objeto Field a partir de uno RecordSet, es decir, para crear un objeto *Field* tendremos que crear primero uno RecordSet.

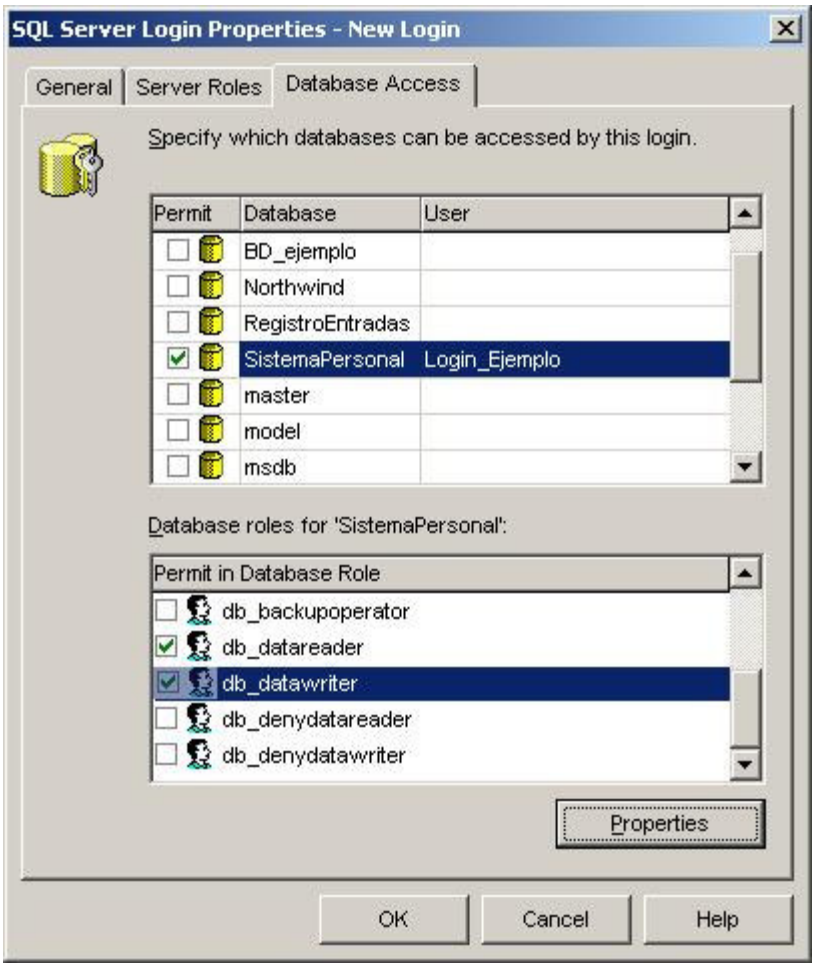


Figura 10.4 Base de datos a la cual estará asociada el login y roles de acceso a esta.

Objeto Connection.

Representa la conexión con una base de datos. Este objeto lo utilizaremos para crear un enlace directo entre nuestra pagina ASP y el servidor de BD. Mientras dure la conexión podremos realizar todas las operaciones que deseemos sobre la base de datos. La conexión terminara cuando lo indiquemos con el método *Close* de este objeto. Este es el objeto más importante, sin el no podemos hacer nada, ya que para trabajar con cualquier elemento de una base de datos, como una tabla o una vista, primero debemos tener una referencia a la BD. Además con este objeto podemos ejecutar sentencias SQL directamente, sin utilizar otro objeto ADO. Para crear este objeto realizamos su instanciacion, mediante la siguiente sentencia:

SET Obj_Conn = Server.CreateObject(ADODB.Connection)

Utilizamos *SET* para asignar a la variable *Obj_Conn* un objeto. Conseguiremos una instancia del objeto Connection con el método *CreateObject* del objeto Server. Indicaremos primero la librería *ADODB* y luego el objeto en concreto que deseamos instanciar: *Connection*.

Este objeto cuenta con diferentes propiedades y métodos para poder configurar su uso, los principales son los siguientes:

- *Propiedad ConnectionString.* Una vez creado el objeto, debemos indicar a que BD queremos que se conecte. Con esta propiedad, pasaremos una cadena de texto al objeto, indicando la base de datos que queremos utilizar, el tipo de base de datos que es y si fuera necesario, el usuario y password de la base.

  Obj_Conn.ConnectionString = "driver={Acces(*.mdb)};dbq=c\BD\base.mdb"

  Si utilizamos un DSN para la conexión, la propiedad anterior no será necesaria de utilizar. Este concepto se explicara mas adelante.
- *Propiedad ConnectionTimeOut.* Sirve para configurar el tiempo de espera de la conexión a la base de datos. Este tiempo, por defecto de 15 segundos, es el que dedica el objeto Connection a intentar conectarse con la base de datos. Si no lo consigue en el tiempo especificado, se produce un error.
- *Método Open.* Este método realiza la conexión real con la base de datos, utilizando los parámetros enviados. También nos permite insertar directamente la propiedad ConnectionString, con lo cual podemos ejecutarlo después justo después de crear el objeto Connection y olvidarnos del resto de propiedades. La s formas de utilizar este método es como sigue:
  1. Después de darle un valor a ConnectionString: Onj_Conn.Open
  2. Sin utilizar ConnectionString, incluimos esta propiedad dentro de Open
     Obj_Conn.Open "driver={acces(*.mdb)};dbq=c:\BD\base.mdb"
  3. Utilizando un DSN, el cual es creado dentro del ODBC y el cual se explico en el capitulo anterior:
     Obj_Conn.Open = "Alumnos;User ID=usuario;Password=password"
- *Método Execute.* Nos permite ejecutar una sentencia SQL. Por ejemplo podríamos consultar información
  Obj_Conn.Execute "SELECT * FROM Alumnos WHERE ciudad = 'Pachuca'"
- *Método Close.* Cierra la conexión con la base de datos, con lo que el objeto Connection se queda disponible para ser utilizado con otra base.
- *Destrucción del objeto.* Nos sirve para liberar memoria y perder todo vinculo con la base de datos: SET Obj_Conn = Nothing

Objeto RecordSet.

Representa una tabla de datas. En este objeto se almacenan las consultas realizadas a la base de datos a la que estemos conectados. Estará formado por filas (registros) y columnas (campos) a los que podremos acceder para exponer la información requerida. Para utilizar este objeto, primero debemos de haber creado un objeto Connection, ya que este será quien indique al RecordSet de que base de datos tiene que coger las tablas. El objeto RecordSet apuntara al primer registro de la tabla consultada. A través de sus métodos y propiedades, podremos ir recorriendo el resto de los registros y consultando sus campos. Este objeto también nos permite modificar, borrar y agregar registros dentro de la tabla seleccionada.

Como en el objeto anterior, veamos las propiedades y métodos mas importantes de este

objeto en el mismo orden en el que se utilizarían en una pagina Web:

- *Instanciacion.* Es muy similar a la del objeto Connection, simplemente cambiamos el nombre del objeto a instanciar:

  SET Obj_RS = Server.CreateObject("ADODB.RecordSet")

- *Colección Fields.* Esta colección contendrá los campos que tiene un registro. Como cosas mas destacables tenemos la posibilidad de conocer el valor de cada campo de la consulta, lo podemos utilizar de la siguiente forma:

  Obs_RS.Fields(No._Campo).Name o Obj_RS.Fields(Nombre_Campo).Name

- *Propiedad RecordCount.* Esta propiedad devuelve el número de registro que contiene el RecordSet.
- *Propiedad EOF.* Devuelve el valor de TRUE cuando hemos llegado al último registro del RecordSet.
- *Método MoveFirst.* Este método nos llevara al primer registro del RecordSet.
- *Método MoveLast.* Con este método nos moveremos hasta el último registro del RecordSet.
- *Método MoveNext.* Nos mueve un registro hacia delante.
- *Método MovePrevious.* Nos mueve un registro hacia atrás.
- *Método GetRows.* Devuelve un arreglo (Array) con todos los registros.
- *Método Close.*Cierra el RecordSet, esto hace que desaparezcan todos los registros del mismo. Una vez ejecutado este método podremos volver a utilizar el RecordSet para otra selección.
- *Destrucción del Objeto.* Se utiliza la misma sentencia que para el objeto Connection:
  SET Obj_RS = Nothing

Objeto Command.

Representa un comando SQL. Con este objeto podemos ejecutar sentencias SQL sobre la base de datos a la cual estemos conectados y almacenar su resultado en un RecordSet.

Al igual que los objetos anteriores, veamos las propiedades y métodos más importantes de este objeto:

- *Propiedad ActiveConnection.* Asignaremos a esta propiedad el objeto Connection que hemos creado anteriormente. Esta es la forma de decirle al objeto Command la base de datos con la cual va a trabajar:
  Obj_Comm.ActiveConnection = Obj_Conn

- *Propiedad CommandText.* Es una cadena de texto con el comando que será ejecutado. La sintaxis para la utilización de esta propiedad es la siguiente:

  Obj_Comm.CommandText = "SELECT * FROM nom_tabla"

- *Método Execute.* Este método sirve para ejecutar el comando almacenado en la propiedad CommandText, su sintaxis es la siguiente:

  ObjComm.Execute

Veamos a continuación un ejemplo completo utilizando los tres objetos anteriormente definidos para establecer conexión a la base de datos desde una pagina ASP:

```
<%
'Ejemplo para conectarnos a base de datos.
Dim Conexion, Comando, RecordSQL, usuario,direccion,sexo
'Creamos el objeto Connection.
Set Conexion = CreateObject("ADODB.Connection")
'Ejecutamos el DSN para conectarnos a la base de datos.
Conexion.Open = DSN=Sistema_Personal;UserID=Personal;Password=Personal;Database=SistemaPersonal"
'Creamos el objeto Command.
Set Comando = CreateObject("ADODB.Command")
'Creamos el objeto RecordSet.
Set RecordSQL = CreateObject("ADODB.RecordSet")
'Utilizamos la propiedad CommandText para ejecutar la sentencia de SQL.
Comando.CommandText = "SELECT
nombre,apellido_paterno,apellido_materno,calle,colonia,municipio,fecha_nacimiento,sexo,estado_civil FROM
usuarios"
'Le asignamos la conexión al objeto Command.
Comando.ActiveConnection = Conexion
'Ejecutamos el objeto Command y lo guardamos en el objeto RecordSet.
Set RecordSQL = Comando.Execute
%>
```

[Vease Anexo A21]

El resultado que se mostraría en la ventana del explorador de la página anterior, se muestra en la figura 10.5.

10.5. Comandos Básicos.

Veamos los comandos básicos utilizados en el lenguaje SQL, que nos permiten generar consultas para ordenar, filtrar, modificar, insertar y extraer información de la base de datos. Como se dijo en la introducción del capitulo, es necesario y responsabilidad del lector aprender la utilización del lenguaje SQL para su buena utilización y entendimiento en el uso con ASP.

Las principales consultas y el modo sencillo de utilización de estas en el lenguaje son las siguientes:

- ***Consulta de Selección SELECT.*** Esta sentencia selecciona registros de una base de datos. Podemos elegir los registros que queremos seleccionar a través de sus distintas variantes (utilizando la cláusula WHERE por ejemplo) y obtener todos los campos de una tabla (utilizando el asterisco *) o solo los que necesitemos. La sintaxis para la utilización de esta sentencia es la siguiente:

  *SELECT * FROM tabla*

o
SELECT campos FROM tabla

En donde *campos* es la lista de campos que se desean recuperar y *tabla* es el origen de donde vamos a obtener los mismos.

- ***Consulta de Acción INSERT INTO.*** Esta sentencia sirve para agregar un registro en una tabla, se le conoce también como "Consulta de Datos Añadidos". Esta consulta puede ser de dos tipos: insertar un único registro en una table o insertar en una tabla los registros contenidos en otra. La sintaxis para la utilización de esta sentencia es la siguiente:

  Para insertar un único registro:
  INSERT INTO tabla (campo1,campo2,...,campoN)
  VALUES (valor1,valor2,...valorN)

  Para insertar registros de otra tabla:
  INSERT INTO TablaDestino (campo1,campo2,...,campoN)
  SELECT campo1,campo2,...,campoN FROM TablaOrigen

- ***Consulta de Acción DELETE.*** Esta sentencia elimina los registros de la tabla indicada. Esta consulta elimina los registros completos, es decir, que no es posible eliminar el contenido de un campo en específico. La sintaxis para la utilización de esta sentencia es la siguiente:

  DELETE tabla WHERE criterio

  Una vez que se han eliminado los registros utilizando esta sentencia, no es posible deshacer la operación. Si desea saber que registros se eliminaran, primero examine los resultados utilizando la consulta de selección que haga uso del mismo criterio y después ejecute esta consulta. Mantenga copias de seguridad de sus datos en todo momento, así si elimina los registros equivocados podrá recuperarlos desde dichas copias.

- ***Consulta de Acción UPDATE.*** Esta sentencia realiza una consulta de actualización que cambia los valores de los campos de una tabla indicada basándose en un criterio específico. La sintaxis para la utilización de esta sentencia es la siguiente:

  UPDATE tabla SET campo1=valor1, campo2=valor2,...,campoN=valorN
  WHERE criterio

  Si en esta sentencia omitimos la cláusula WHERE, todos los registros de la tabla especificada serán actualizados, por lo que hay que tener especial cuidado en utilizarla si solo se van a actualizar algunos registros.

Las consultas que acabamos de ver son las básicas que utilizaremos para manipular la información de la base de datos, utilizando las diferentes cláusulas que nos ayudan a particularizar las operaciones.

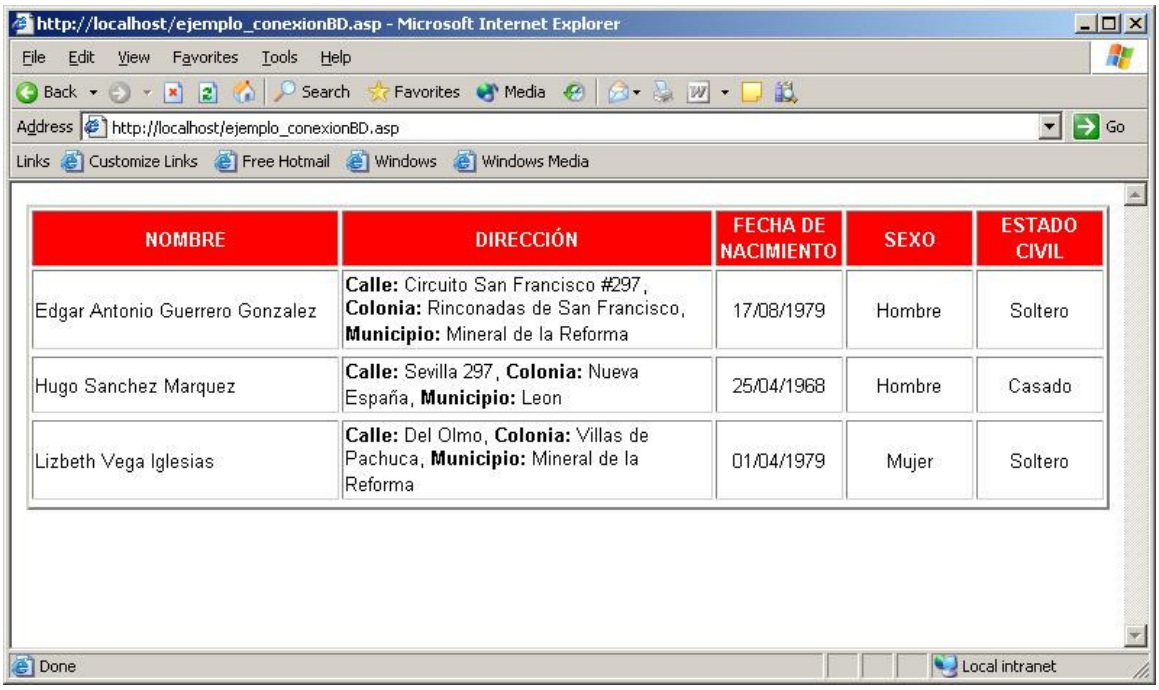
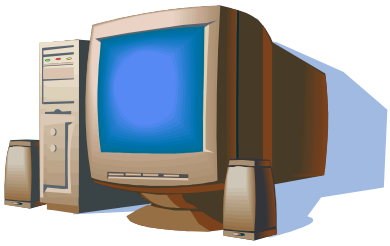


Figura 10.5 Ejemplo de cómo conectarnos a base de datos y obtener la información existente o requerida.

CAPITULO 11

Caso de Estudio: Tralcom S.A. de C.V.



En este capítulo se aplicarán los puntos desarrollados anteriormente para ofrecer un ejemplo práctico de cómo se desarrolla una aplicación con ASP. Cabe señalar que se tocaran todos los temas de esta monografía, sin embargo, no podré llegar a un nivel más amplio de detalle del funcionamiento interno del sistema presentado por ser propiedad de la empresa donde laboro y ser una marca registrada.

11.1. Introducción.

Tralcom es una empresa 100% mexicana, dedicada al desarrollo de sistemas para educación a distancia, con lo cual ofrece soluciones de educación de manera remota, ayudando con esto a la educación y/o capacitación de las personas, sin necesidad de que estas se muevan de su lugar de origen.

Los productos de Tralcom han sido diseñados para resolver diversas necesidades en las organizaciones, tomando como base 3 tendencias de servicios que se han generado en el mundo de la tecnología sobre Internet:

E-LEARNING

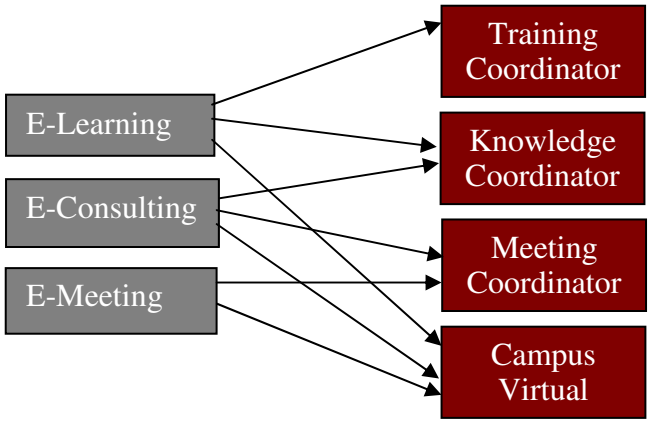
- Que consiste en la capacidad de entregar capacitación por medio de elementos tecnológicos como: Videoconferencia, Entrenamiento basado en Web (WBT) y elementos multimedia que integren contenido didáctico y asimilable por los usuarios.
- Esta herramienta permite montar contenido y dar servicios de asesoría para la capacitación de las empresas y las Universidades o instituciones educativas.

E-CONSULTING

- Que consiste en la utilización de herramientas para facilitar el contacto entre el consultor y los clientes, que le permita a la organización de consultores trabajar en forma colaborativa y que el esfuerzo general de trabajo tenga una mezcla de trabajo en campo y de trabajo remoto, que le permita al grupo de consultores atender a mas clientes en menor tiempo.

E-MEETING

- Que consiste en el trabajo colaborativo de toda la organización para lograr un esquema de seguimiento y trabajo en equipo eliminando las barreras geográficas de la red Cetro.
- Tralcom promueve soluciones que permiten a las organizaciones consolidar estas tres disciplinas a través de la utilización de sus herramientas:



TRAINING COORDINATOR©

Training Coordinator© es toda la infraestructura necesaria para montar una Universidad Virtual. Es una herramienta que permite un control total de una universidad virtual, desde administrar el plantel virtual, hasta publicar y controlar el acceso por Internet a los diferentes cursos de sus planes de estudio.

Las grandes y pequeñas entidades educativas se enfrentan a la necesidad de modernizarse y permanecer competitivos en el mercado, incorporar servicios extramuros por Internet es de vital importancia para garantizar a sus estudiantes todos los medios modernos de aprendizaje.

- Cada día más hogares y empresas están conectados a Internet, un estudiante o profesionista que desee estar actualizados debe conectarse a Internet.

La tecnología necesaria para implementar un "campus virtual" era extremadamente cara y compleja, Training Coordinator© ofrece una solución empaquetada y de bajo costo para lograrlo rápidamente

Desarrollado con tecnología 100% Web (utilizando el lenguaje ASP para el desarrollo de las paginas), Training Coordinator incorpora lo último en tecnología para facilitar la experiencia del usuario con la aplicación permitiendo un ambiente de utilización y personalización sencillo e intuitivo. Mediante la explotación de la tecnología de Digital Dashboards permite incorporar partes Web (Webparts) que le ofrecen integrar en un solo portal acceso a sitios de interés, herramientas de búsqueda y los nuevos componentes Web que Microsoft Corporation esta desarrollando como: Office XP Web Parts, MSN Encarta Referente, buscadores, etc. Así como muchísimos Webparts que se están ofreciendo en el mercado para ser fácilmente integrados en el portal de Training Coordinator.

11.2. Base de Datos.

Ya vimos en el capitulo 10 como crear una base de datos, a continuación veremos lo que debemos hacer para utilizar la base de datos de Training Coordinator©:

1. Crear una BD en SQL Server con el nombre **TrainingCoord_XX** utilizando la siguiente sintaxis:

```
Use master
CREATE DATABASE [TrainingCoord_XX]
ON(
NAME = N'TrainingCoord_DU1_Data',
FILENAME=N'C:\ProgramFiles\MicrosoftSQLServer\MSSQL\Data\TrainingCoord_XX_Data.MDF',
SIZE = 253,
FILEGROWTH = 10%
)
LOG ON(
NAME = N'TrainingCoord_DU1_Log',
FILENAME=N'C:\ProgramFiles\MicrosoftSQLServer\MSSQL\Data\TrainingCoord_XX_log.LDF',
SIZE = 1441,
FILEGROWTH = 10%
)
```

2. Restaurar el Backup que se encuentra en la carpeta [**Unidad de disco de la aplicación**]***:\TrainingCoordinator\BD\TrainingCoord_XX_bkp.bkp*** a la base de datos **TrainingCoord_XX**, creada en el paso 1.

3. Una vez restaurada la base de datos de Training, crear un Login(usuario) a nivel general con las siguientes características:

    a. Name: **training_XX**
    b. Password: **tnttc_XX**
    c. Database: **TrainingCoord_XX**
    d. Language: **Default**
    e. Database Access: **TrainingCoord_XX**
    f. Permit in Database Rol: **public**, **bd_owner**, **db_datareader**, **db_datawriter** sobre la **BD TrainingCoord_XX**

4. Validar que el usuario y password sean correctos, lo anterior se puede hacer entrado a la aplicación **QueryAnalizer (Analizador de Consultas)** de **SQL Server** tecleando el usuario y password que se dio de alta en el paso 3(vease figura 11.1), si esta herramienta permite el acceso al usuario, significa que el usuario fue creado correctamente:

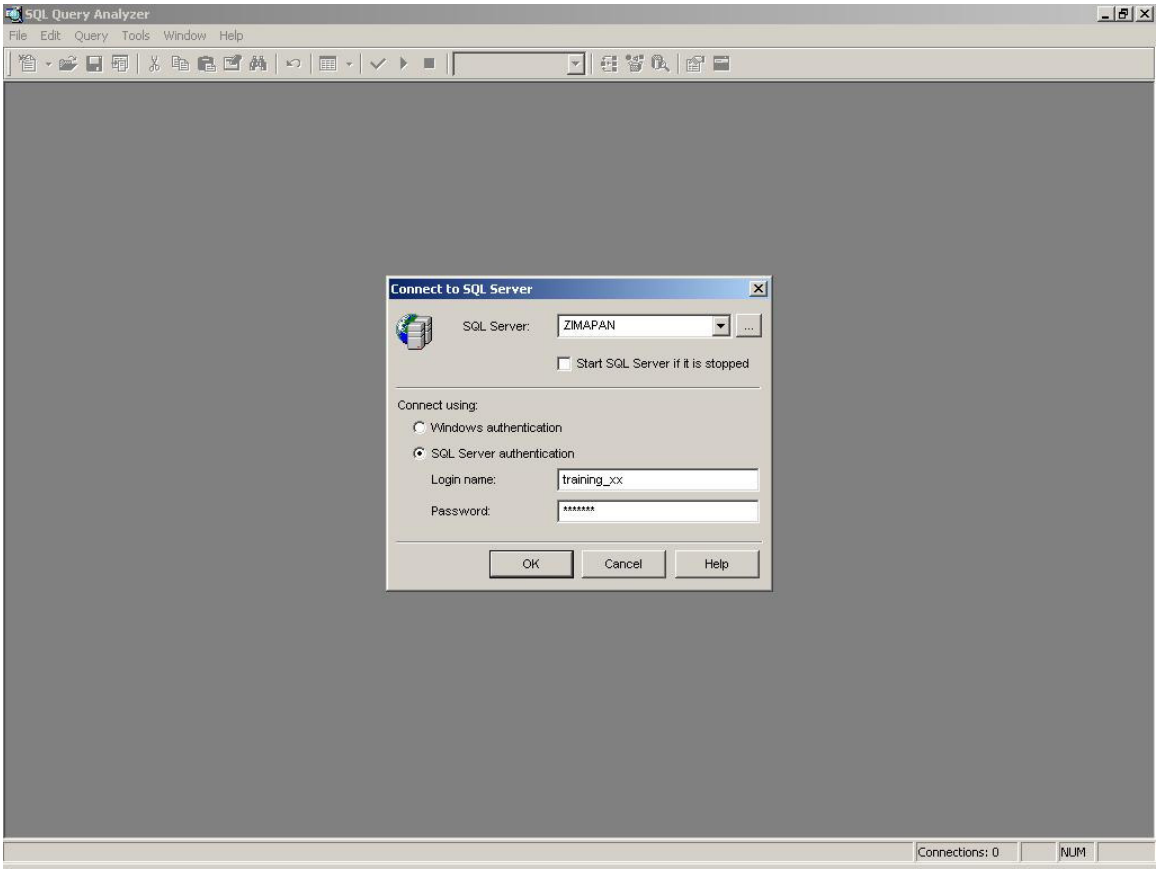


Figura 11.1 Verificación del Acceso del usuario training_xx a la base de datos desde el Query Analyser.

11.3. Configuraciones del IIS.

Para levantar el servidor y poder publicarlo para el acceso a este, hacemos los siguientes pasos de configuraciones en el IIS:

1. Crear el **Web Site Cliente** en el **IIS** de Windows. con las siguientes características:
   a. Alias: nom_cliente
   b. Directory: Se debe especificar la ruta al directorio Root, [Unidad de disco de la aplicación]:\TrainingCoordinator\TC\Root
   c. Acces Permissions: Read, Run_Script (such as ASP)
   d. Por ultimo se debe poner como página inicial default.asp en las propiedades del sitio, así como seleccionar únicamente la opción Enable anonymous access, que se encuentra en la pestaña Directory Security, Authentication and Access control.
2. Se debe crear un directorio virtual dentro del Web Site Cliente, para el acceso al sitio de training, con las siguientes características:
   a. Alias: tc-nom_cliente
   b. Directory: Se debe especificar la ruta al directorio tc-emp-xx, [Unidad de disco de la aplicación]:\TrainingCoordinator\TC\Root\tc-emp-xx
   c. Acces Permissions: Read, Run_Script (such as ASP)
   d. Por ultimo se debe poner como página inicial default.asp en las propiedades del sitio, así como seleccionar únicamente la opción Basic Autentification, que se encuentra en la pestaña Directory Security, Authentication and Access control.

3. Se debe crear un directorio virtual dentro del Web Site Cliente, para el acceso al sitio de inscripciones de training, con las siguientes características:
   a. Alias: tc-insc-nom_Cliente
   b. Directory: Se debe especificar la ruta al directorio tc-insc_emp, [Unidad de disco de la aplicación]:\TrainingCoordinator\TC\Root\tc-insc-emp-xx
   c. Acces Permissions: Read, Run_Script (such as ASP)
   d. Por ultimo se debe poner como página inicial default.asp en las propiedades del sitio, así como seleccionar únicamente la opción Enable anonymous access, que se encuentra en la pestaña Directory Security, Authentication and Access control.

11.4. Registro del ODBC.

Como vimos en el capitulo 9, creamos una conexión de ODBC (**System DNS**) que apunte a la **BD TrainingCoord_XX** para poder accesar a la información de la misma(vease figura 11.2), para ello se deben hacer los siguientes pasos:

a. Driver: **SQL Server**
b. Nombre del DNS: **TrainingCoord_XX**
c. Descripción: **TrainingCoord_XX**
d. Server: Nombre del servidor donde reside la **BD**
e. Usuario: **training_XX**
f. Password: **tnttc_XX**

Una vez terminada la creación del DSN para la base de datos de TrainingCoord_XX, ya podremos accesar a la información de la misma mediante dicho DSN.

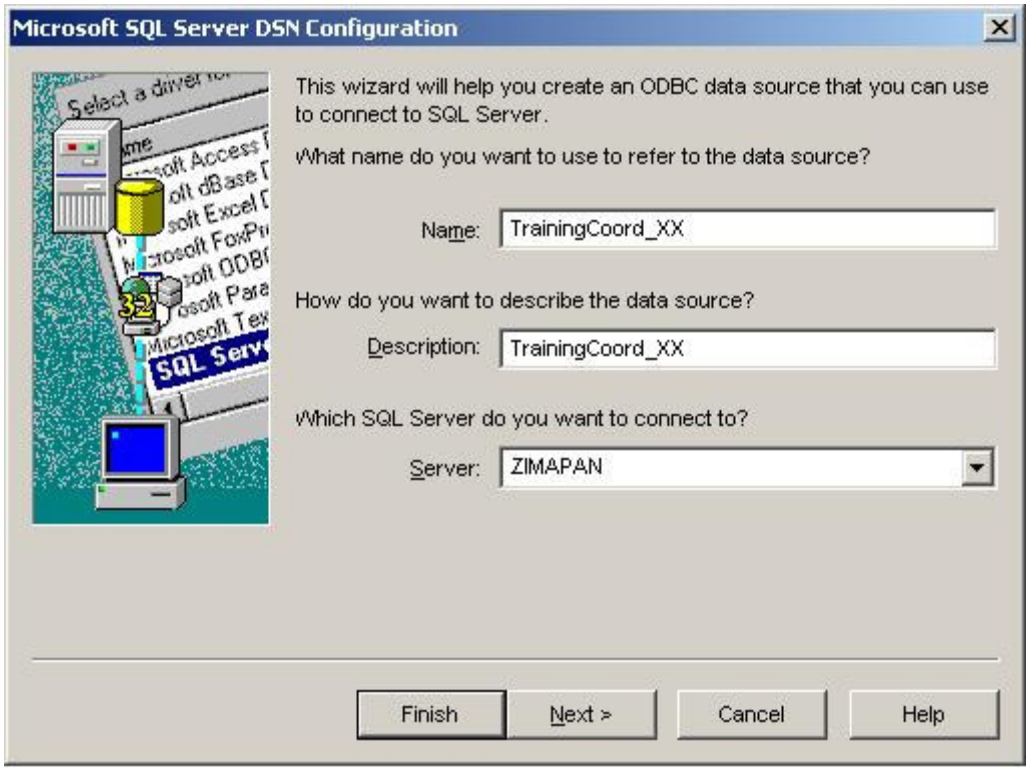


Figura 11.2 Creación del DSN para el acceso a la base de datos de Training.

11.5. Inicialización del archivo Global.asa.

La configuración del archivo Global.asa, que se encuentra en la ruta [Unidad de disco de la aplicación]:\TrainingCoordinator\TC\Root\tc-emp-xx\global.asa, debe realizarse inicializando los eventos descritos en el capitulo 8 para este archivo, dichos eventos son los que servirán para poner en funcionamiento el sitio(Vease Anexo A22).

Una vez modificado el archivo global.asa y actualizada la tabla de donde se obtienen los datos de configuración, es necesario reiniciar el IIS para que la aplicación tome los cambios de manera inmediata.

11.6. Codificación de páginas ASP del sistema.

Ya vimos a lo largo de este trabajo las sintaxis de la declaración de algunas sentencias en ASP, así como ejemplos del uso de las mismas, veamos ahora la utilización de algunas de estas sentencias dentro del sistema Training Coordinator y su enorme funcionalidad en el desarrollo de páginas Web.

Para poder administrar la información de los usuarios del sistema, utilizamos el lenguaje SQL para poder obtener la información requerida, para esto utilizamos un query como el que se muestra a continuación:

select a.id_usuario,a.usuario,password,last_login,count_login,situacion,isnull(ape_pat,' ')+' '+ isnull(ape_mat,' ')+' '+isnull(nombre1,' ')+' '+isnull(nombre2,' ')+' ' ,email, ape_pat, ape_mat, nombre1, nombre2,usuario_alfa

```
from trc_usuarios a, trc_usuario_division b
where situacion='A'
and a.usuario = b.usuario
and b.division = 1
order by a.usuario
```

En este query podemos ver que ligamos dos tablas: la que tiene la información de los usuarios y la que indica a que division pertenecen dichos usuarios, ya que podría darse el caso de que pertenezcan a varias divisiones de la empresa o escuela (por ejem: división Hidalgo, Chihuahua, San Luís Potosí, etc.) y le indicamos los usuarios de la division que queremos obtener.

Una vez que el query nos devuelve la información solicitada, en la página ASP creamos la conexión y obtenemos el RecordSet donde tendremos toda la información, tomamos este recordset (que lo llamamos rsig) y empezamos a pintar la información en nuestra página, para esto utilizamos el bucle Do…Loop (el cual vimos en el capitulo 4):

```
Do While Not rsig.EOF
<%
'Utilizamos la función Len y Mid de VBScript para cortar el nombre del usuario cuando este se mas largo de 90
caracteres y así que no sobrepase el tamaño de la página.
if len(rsig(6)) > 90 then
nombre = mid(rsig(6),1,90) & " " & mid(rsig(6),91,len(rsig(6)) - 90)
end if
if len(rsig(6)) > 170 then
nombre = mid(nombre,1,170) & " " & mid(nombre,171,len(nombre) - 170)
end if
if len(rsig(6)) > 250 then
nombre = mid(nombre,1,250) & " " & mid(nombre,251,len(nombre) - 250)
end if
if len(rsig(6)) <= 90 then
nombre = rsig(6)
end if
%>
<td width="14%" align="left">
<font face="tahoma" size="1" color="<%=session.contents("color_text")%>"><%=rsig(12)%></font>
</td>
<td align="left">
<font face="tahoma" size="1" color="<%=session.contents("color_text")%>"><%=nombre%></font>
</td>
</tr>
<tr>
<td colspan="13"><hr color="<%=session.contents("color_text")%>"></td>
</tr>
<%
rsig.movenext
Loop
%>
```

Como podemos observar en el código anterior, no es necesario tener que codificar en html todas la líneas de los usuarios existentes en la base de datos, basta con codificar una sola línea y con el bucle Do…Loop se generaran dinámicamente las líneas necesarias para los usuarios obtenidos en la consulta, lo cual nos evita tener que duplicar código y hace que nuestras paginas sean dinámicas y sencillas de manejar. El resultado que se obtiene en la ventana de nuestro navegador puede verse en la figura 11.3.

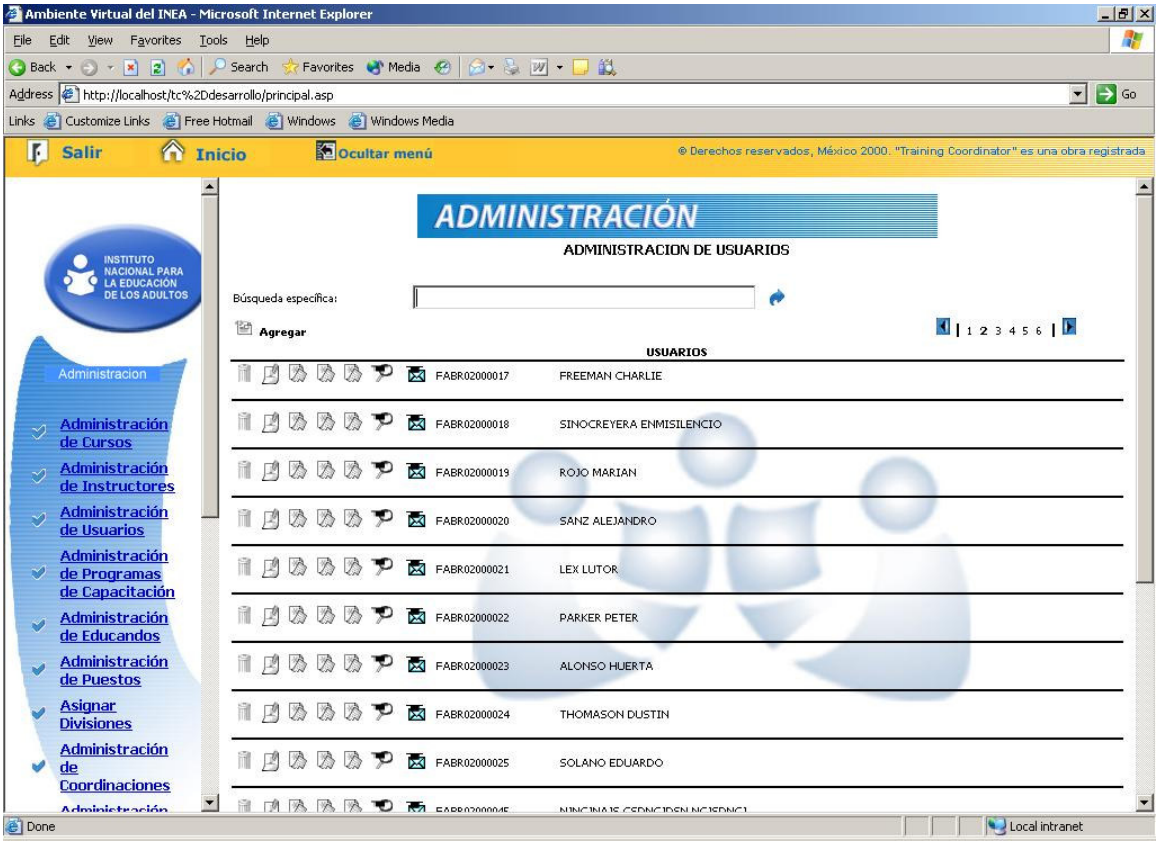


Figura 11.3 Pintado de la información de los usuarios almacenados en la base de datos.

Continuando con la manipulación de los usuarios, vamos a ver ahora el ejemplo del uso de formularios (esto se explico en el capitulo 6). En este ejemplo tenemos un formulario que utiliza el método "post" para el manejo de la información, y en el cual se piden los datos del usuario que se va a dar de alta(ver figura 11.4).

Una vez que el usuario llena todos los datos solicitados y da click en el botón "Alta de Usuario", se ejecuta el proceso de envío de información del formulario por el método post, donde se hace ejecuta el "action" del formulario a la misma pagina para insertar en la base de datos el usuario nuevo, y al regresar a la administración general de usuarios ya podremos ver al usuario que acabamos de dar de alta listado en nuestra pagina.

Por ultimo veamos la aplicación de los objetos integrados que se explico en el capitulo 7 de este trabajo. Sigamos entonces utilizando la administración de usuarios para ilustrar lo anterior, entremos entonces a la parte de la modificación de usuarios. Para poder saber que usuario en especifico debemos de recuperar de la base de datos, necesitamos la clave del usuario, entonces desde la página de administración de usuarios enviamos a la de modificación la clave del usuario que queremos modificar, dicha clave al ser única, evitara que tengamos el conflicto de que exista mas de un usuario con la misma clave en el sistema, como dicha clave es enviada de una página a la otra mediante URL, la forma de recuperarla en la página de modificación es de la siguiente manera:

```
usuario = request.QueryString("usuario")
usuario_alfa = request.QueryString("usu_alfa")
```

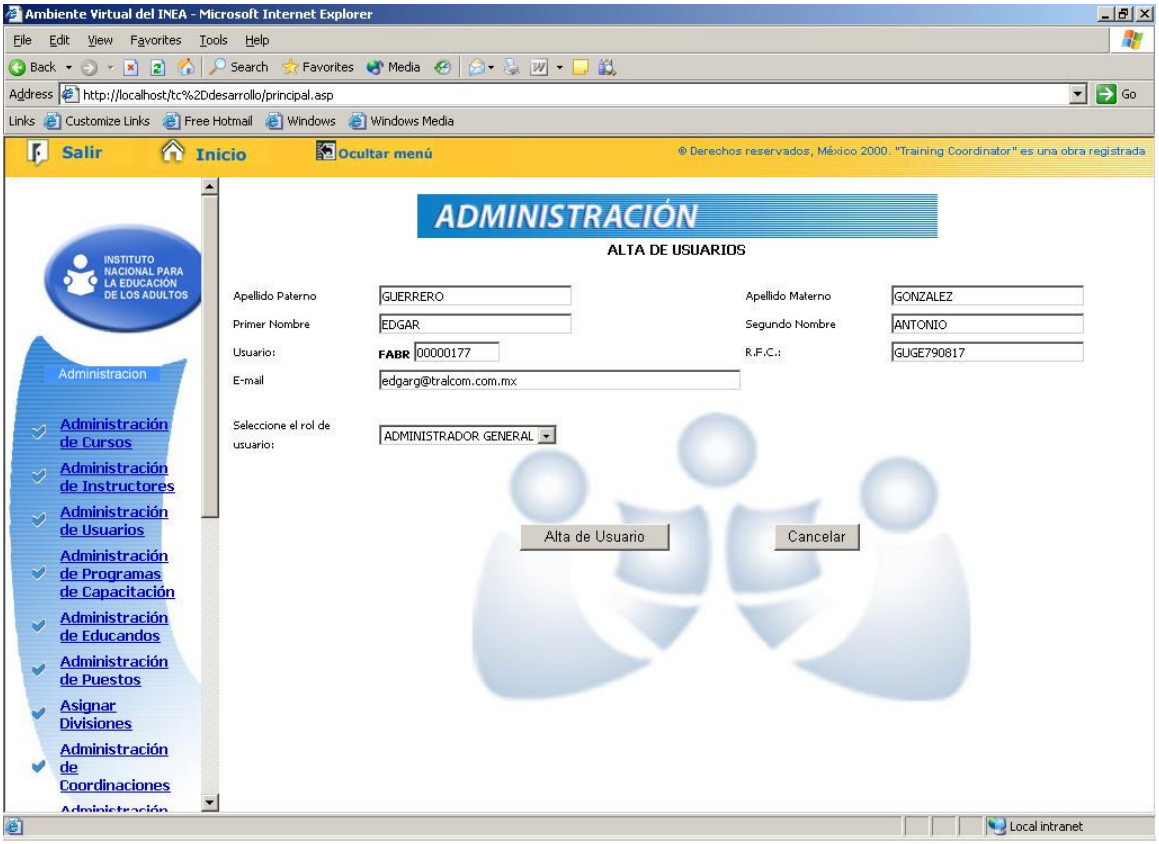


Figura 11.4 Alta de Usuarios en el sistema.

De esta forma, al construir la consulta de SQL que obtiene los datos del usuario, utilizamos la variable usuario para realizar dicha consulta y traer los datos correctos, de esta forma nuestra consulta a la base de datos es dinámica:

```
select id_usuario, usuario,
isnull(ape_pat,' '), isnull(ape_mat,' '), isnull(nombre1,' '), isnull(nombre2,' '), isnull(email,' '),usuario_alfa
from   trc_usuarios
where  usuario = '" & usuario & "' "
and    situacion = 'A'
```

Una vez que obtenemos los datos necesitados, almacenamos en variables los datos para después mostrarlos en la página para que puedan ser modificados:

```
ape_pat = rsig(2)
ape_mat = rsig(3)
nom1 = rsig(4)
nom2 = rsig(5)
email = rsig(6)
```

[Vease Anexo A23]

Como podemos ver en el código anterior, las variables que se crean con los datos de la consulta, son utilizadas para rellenar el formulario de modificación y de esta manera poder ver los datos que tiene ese usuario y así cambiar los que se necesiten modificar. También podemos ver que se utiliza el objeto session, ya que el color de la letra esta definido en variables de sesion, con lo cual se evita poner de manera “fija” el color que se

va a poner en cada una de las páginas. El resultado que se vería en el navegador lo podemos observar en la figura 11.5

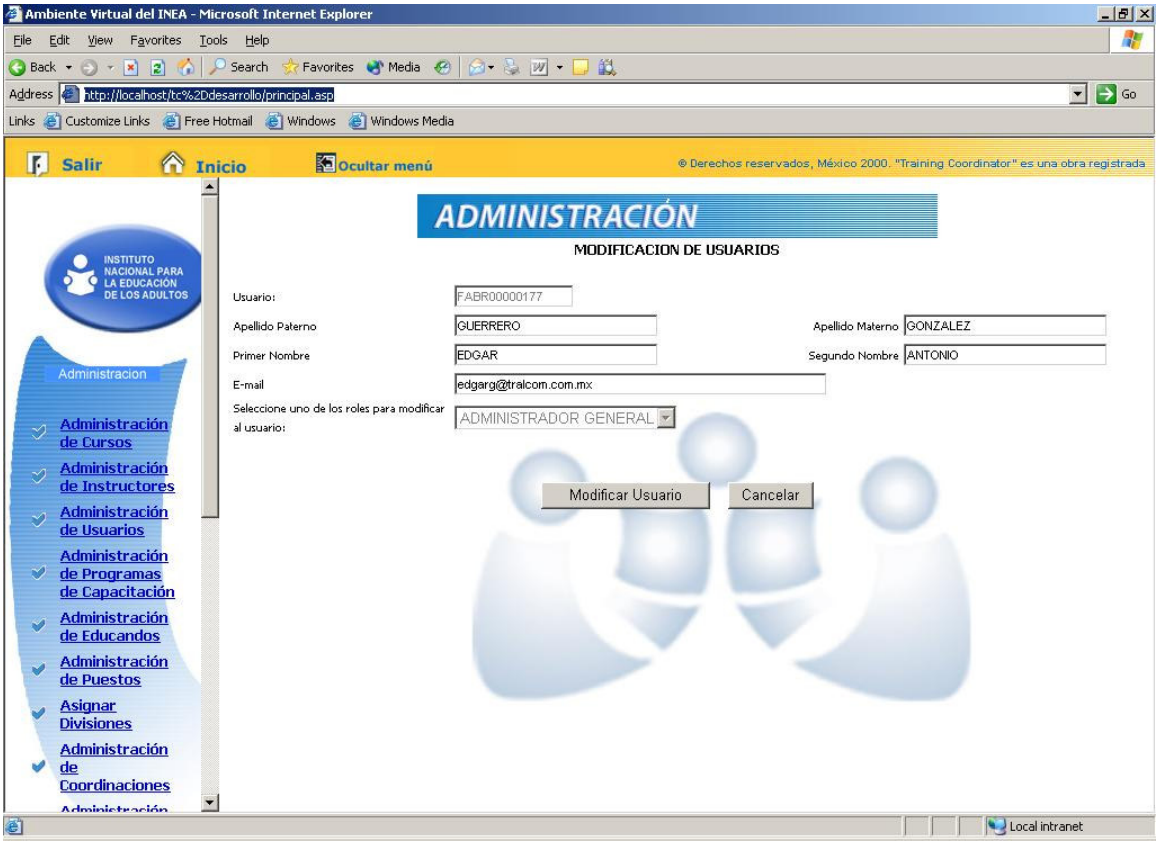


Figura 11.5 Modificación de los datos del usuario seleccionado.

Posteriormente ya realizamos el proceso de envío de datos por formulario (igual que en el alta de usuarios) para realizar la modificación de los datos del usuario y almacenarlos en la base de datos.

CONCLUSIONES

La programación de páginas Web en ASP, tiene un enorme auge entre los desarrolladores, por su enorme facilidad y potencia en el desarrollo con esta tecnología, ya que en el caso de la programación en HTML, se debe de realizar de forma estática, haciendo que la programación sea tediosa y nada modular.

La posibilidad de mezclar las dos tecnologías anteriores, permite que tengamos un lenguaje donde podemos aprovechar la facilidad y ventajas del HTML, con la potencia del ASP para tener así un lenguaje de programación sencillo y a la vez muy poderoso para el desarrollo de aplicaciones Web dinámicas.

Por otra parte, la programación en HTML ha sido utilizada por los programadores desde hace ya bastante tiempo, teniendo el desarrollo de sitios sencillos y que tenían muy poca interacción con los usuarios del sistema. Sin embargo, con el auge de la programación en ASP, estos sitios se han podido realizar en forma dinámica e interactiva con los usuarios del Web, haciendo dichos sitios más interesantes y útiles para todos los usuarios del sistema..

Este tipo de programación, por el tamaño de las empresas actuales, las hace mucho mas competitivas y atractivas para todos sus clientes, por tener sistemas fáciles y muy poderosos de utilizar, esto a las compañías también les significa un enorme ahorro en la inversión del desarrollo de los sistemas, ya que evita la plana y poco efectiva programación estructurada que se realizaba en el pasado.

En síntesis, la funcionalidad en ASP es algo realmente revolucionario dentro del mundo de la programación, pero no nuevo, ya que existen desde hace mucho tiempo tecnologías mas rudimentarias, concretamente, se usaba la tecnología CGI(Common Gateway Interface – Pasarela de Interface Común), que básicamente son programas independientes, escritos en cualquier lenguaje de programación(Pascal, C, Fortran, etc.), los cuales realizan mandatos para generar una salida HTML, sin embargo, ASP es un lenguaje de programación que vino a sustituir estos programas, por ser un lenguaje muy potente que ofrece un enorme rendimiento a los usuarios y desarrolladores.

BIBLIOGRAFÍA

- **[A] ASP 3.0. INICIACIÓN Y REFERENCIA**
  Alejandro Alcocer Jarabo; Jesús Bobadilla Sancho; Luís Rodríguez-Manzaneque Sánchez
  Editorial McGraw-Hill
  Año: 2001

- **[B] ASP. SIN ERRORES**
  Derek Ferguson
  Editorial McGraw-Hill
  1ra. Edición
  Año: 2001

- **[C] CREACION DE APLICACIONES WEB EN WINDOWS NT, ACTIVE SERVER PAGES**
  Jesús Bobadilla; Alejandro Alcocer
  ALFAOMEGA

- **[D] ACTIVE SERVER PAGES 3 - CREACION DE APLICACIONES WEB A TRAVÉS DE EJEMPLOS.**
  Jesús Bobadilla, Alejandro Alcocer, Luís Rodríguez – Manzaneque
  ALFAOMEGA/RA-MA
  Año: 2000

- **[E] FUNDAMENTOS DE PROGRAMACIÓN EN ASP 3.0**
  Mercer
  Editorial McGraw-Hill

- **[F] PROGRAMACIÓN CON ASP 3**
  Jorge Serrano Pérez
  Ed. Anaya Multimedia
  1ra. Edición
  Año: 2000

REFERENCIAS ELECTRÓNICAS

- **[1] URL:** http://www.desarrolloweb.com/manuales/8/
  Autor: Miguel Ángel Álvarez, Rubén Álvarez.
  Organización: Guiarte Multimedia S.L.
  Fecha de consulta: 10 de octubre del 2005.

- **[2] URL:** http://www.webestilo.com/asp/
  Autor: Joaquín Gracia Murugarren.
  Fecha de consulta: 15 de octubre del 2005.
  País: España.
  Año: 1998 - 2004

- **[3] URL:** http://www.asptutor.com/asp/ejemplosdecodigo.asp
  Autor: Pedro Rufo Martín.
  Fecha de consulta: 10 de noviembre del 2005.
  Año: 2001 - 2005

- **[4] URL:** http://www.soloasp.com.ar/articulo.asp
  Fecha de consulta: 10 de octubre del 2005
  Pais: Argentina

- **[5] URL:** http://www.aspfacil.com/articulos/default.asp?cat=3orden=c
  Autor: Guido Laghi.
  Fecha de consulta: 5 de diciembre del 2005.
  País: Argentina.
  Año: 2001 - 2005

- **[6] URL:** http://www.htmlweb.net/asp/asp.html
  Fecha de consulta: 20 de octubre del 2005

- **[7] URL:** http://www.portalvb.com/CursosASPAp.asp?Ap=0000
  Fecha de consulta: 20 de diciembre del 2005.

- **[8] URL:** http://www.webtaller.com/manual-asp/manual_asp.php
  Organización: Factoría de Internet S.L
  Fecha de Consulta: 14 de Noviembre del 2005
  Pais: España
  Año: 2003 - 2006

ANEXO A

[A1]

```
<%@LANGUAGE="VBSCRIPT"%>
<%Option Explicit%>
<%
   Dim resultado, variable1, variable2
   Variable1 = 8
   Variable2 = 6
%>
<HTML>
<HEAD>
  <TITLE>EJEMPLO DE OPERADORES ARITMETICOS</TITLE>
</HEAD>
<BODY>
<%
Response.Write("Ejemplo de Operadores Aritméticos:") & "<BR><BR>"
Response.Write("Variable1: " & variable1) & "<BR>"
Response.Write("Variable2: " & variable2) & "<BR><BR>"
Response.Write("Operador Suma: Variable1+Variable2 = " & variable1 + variable2) & "<BR>"
Response.Write("Operador Resta: Variable1-Variable2 = " & variable1 - variable2) & "<BR>"
Response.Write("Operador Multiplicación: Variable1*Variable2 = " & variable1 * variable2) & "<BR>"
Response.Write("Operador División: Variable1/Variable2 = " & variable1 / variable2) & "<BR>"
Response.Write("Operador Exponenciación: Variable1^Variable2 = " & variable1 ^ variable2) & "<BR>"
Response.Write("Operador Modulo: Variable1ModVariable2 = " & variable1 Mod variable2) & "<BR>"
Response.Write("Operador División Entera: Variable1\Variable2 = " & variable1 \ variable2) & "<BR>"
Response.Write("Operador Negación: -(Variable1*Variable2) = " & -(variable1 * variable2)) & "<BR>"
Response.Write("Operador Concatenación: Variable1&Variable2 = " & variable1&variable2) & "<BR>"
%>
</BODY>
</HTML>
```

[A2]

```
<%@LANGUAGE="VBSCRIPT"%>
<%Option Explicit%>
<%
   Dim resultado, variable1, variable2
   Dim Igualdad, Desigualdad, Menorque, Mayorque, MenorIgual, MayorIgual, OperadorIs
   variable1 = 8
   variable2 = 6
            Igualdad = variable1 = variable2
            Desigualdad = variable1 <> variable2
            Menorque = variable1 < variable2
            Mayorque = variable1 > variable2
            MenorIgual = variable1 <= variable2
            MayorIgual = variable1 >= variable2
            OperadorIs = IsNull(variable1)
%>
<HTML>
<HEAD>
  <TITLE>EJEMPLO DE OPERADORES DE COMPARACIÓN</TITLE>
</HEAD>
<BODY>
<%
 Response.Write("Ejemplo de Operadores de Comparación:") & "<BR><BR>"
 Response.Write("Variable1: " & variable1) & "<BR>"
 Response.Write("Variable2: " & variable2) & "<BR><BR>"
 Response.Write("Operador Igualdad: Variable1 = Variable2 = " & Igualdad) & "<BR>"
 Response.Write("Operador Desigualdad: Variable1 <> de Variable2 = " & Desigualdad) & "<BR>"
 Response.Write("Operador Menor o igual que: Variable1 <= Variable2 = " & MenorIgual) & "<BR>"
```

```
Response.Write("Operador Mayor o igual que: Variable1 >= Variable2 = " & MayorIgual) & "<BR>"
Response.Write("Operador Mayor que: Variable1 > Variable2 = " & Mayorque) & "<BR>"
Response.Write("Operador Menor que: Variable1 < Variable2 = " & Menorque) & "<BR>"
Response.Write("Operador Is: IsNull(Variable1) = " & OperadorIs) & "<BR>"
%>
</BODY>
</HTML>
```

[A3]

```
<%@LANGUAGE="VBSCRIPT"%>
<%Option Explicit%>
<%
        Dim resultado, variable1, variable2, YLogico, OLogico, OrExclusivo, Equivalencia, Negacion
        variable1 = 8
        variable2 = 6
        YLogico = variable1 = 8 And variable2 = 6
        OLogico = variable1 < 5 Or variable2 >= 6
        OrExclusivo = variable1 <= 8 Xor variable2 <> 6
        Equivalencia = variable1 * 3 = 24 Eqv variable2 * 4 = 24
        Negacion = Not (variable2 < variable1)
%>
<HTML> <HEAD>
  <TITLE>EJEMPLO DE OPERADORES LÓGICOS</TITLE>
</HEAD>
<BODY>
<%
  Response.Write("Ejemplo de Operadores Lógicos:") & "<BR><BR>"
  Response.Write("Variable1: " & variable1) & "<BR>"
  Response.Write("Variable2: " & variable2) & "<BR><BR>"
  Response.Write("Y Lógico: (Variable1=8) And (Variable2=6) = " & YLogico) & "<BR>"
  Response.Write("O Lógico: (Variable1<5) Or (Variable2>=6) = " & OLogico) & "<BR>"
  Response.Write("Or Exclusivo: (Variable1<=8) Xor (Variable2<>6) = " & OrExclusivo) & "<BR>"
  Response.Write("Equivalencia: (Variable1*3=24) Eqv (Variable2*4=24) = " & Equivalencia) & "<BR>"
  Response.Write("Negacion: Not(Variable2 < Variable1) = " & Negacion) & "<BR>"
%>
</BODY></HTML>
```

[A4]

```
<%@LANGUAGE="VBSCRIPT"%>
<%Option Explicit%>
<%
   Dim MayorEdad, MiNombre, MiEdad
   MayorEdad = 18
   MiNombre = "Edgar Antonio"
   MiEdad = 26
%>
<HTML>
<HEAD>
  <TITLE>EJEMPLO DE LA SENTENCIA IF</TITLE>
</HEAD>
<BODY>
<%
  Response.Write("EJEMPLO DEL USO DE LA SENTENCIA IF") & "<BR><BR>"
  Response.Write("Variables: MiNombre, MiEdad, MayorEdad") & "<BR><BR>"
  Response.Write("Nombre: " & MiNombre) & "<BR>"
  Response.Write("Edad: " & MiEdad) & "<BR><BR>"
   If  MiEdad >= MayorEdad Then
     Response.Write("Hola " & MiNombre & "  ya eres Mayor de Edad")
   End If
%>
</BODY>
</HTML>
```

[A5]

```
<%@LANGUAGE="VBSCRIPT"%>
<%Option Explicit%>
<%
   Dim MiNombre, MiSexo
   MiNombre = "Edgar Antonio"
   MiSexo = "M"
%>
<HTML>
<HEAD>
  <TITLE>EJEMPLO DE LA SENTENCIA IF</TITLE>
</HEAD>
<BODY>
<%
 Response.Write("EJEMPLO DEL USO DE LA SENTENCIA IF") & "<BR><BR>"
 Response.Write("Variables: MiNombre, MiEdad, MayorEdad") & "<BR><BR>"
 Response.Write("Nombre: " & MiNombre) & "<BR>"
 Response.Write("Sexo: " & MiSexo) & "<BR><BR>"
 If MiSexo = "F" Then
   Response.Write("Hola " & MiNombre & " eres del sexo Femenino")
 Else
   Response.Write("Hola " & MiNombre & " eres del sexo Masculino")
 End If
%>
</BODY>
</HTML>
```

[A6]

```
<%@LANGUAGE="VBSCRIPT"%>
<%Option Explicit%>
<%
   Dim MiNombre, MiOrigen
   MiNombre = "Edgar Antonio"
   MiOrigen = "Hidalgo"
%>
<HTML>
<HEAD>
  <TITLE>EJEMPLO DE LA SENTENCIA SELECT...CASE</TITLE>
</HEAD>
<BODY>
<%
Response.Write("EJEMPLO DEL USO DE LA SENTENCIA SELECT...CASE") & "<BR><BR>"
Response.Write("Variables: MiNombre, MiOrigen") & "<BR><BR>"
Response.Write("Nombre: " & MiNombre) & "<BR>"
Response.Write("Origen: " & MiOrigen) & "<BR><BR>"
   Select Case (MiOrigen)
            Case "Chiapas"
                     Response.Write("Hola " & MiNombre & " eres Chiapaneco")
            Case "Oaxaca"
                     Response.Write("Hola " & MiNombre & " eres Oaxaqueño")
            Case "Guerrero"
                     Response.Write("Hola " & MiNombre & " eres Guerrerense")
            Case "Yucatán"
                     Response.Write("Hola " & MiNombre & " eres Yucateco")
            Case "Guanajuato"
                     Response.Write("Hola " & MiNombre & " eres del Bajío")
            Case "Querétaro"
                     Response.Write("Hola " & MiNombre & " eres Queretano")
           Case "Hidalgo"
                     Response.Write("Hola " & MiNombre & " eres orgullosamente Hidalguense")
            Case "Sonora"
```

```
                Response.Write("Hola " & MiNombre & " eres Sonorense")
        Case "San Luís Potosí"
                Response.Write("Hola " & MiNombre & " eres Potosino")
        Case "Sinaloa"
                Response.Write("Hola " & MiNombre & " eres Sinaloense")
        Case "Nuevo León"
                Response.Write("Hola " & MiNombre & " eres Norteño")
        Case Else
                Response.Write("Hola " & MiNombre & " tienes un origen desconocido")
  End Select
%>
</BODY>
</HTML>
```

[A7]

```
<%@LANGUAGE="VBSCRIPT"%>
<%Option Explicit%>
<%
  Dim Meses, cont
  Meses = 12
%>
<HTML>
<HEAD>
  <TITLE>EJEMPLO DEL BUCLE FOR...NEXT</TITLE>
</HEAD>
<BODY>
<%
        Response.Write("EJEMPLO DEL USO DEL BUCLE FOR...NEXT") & "<BR><BR>"
        Response.Write("Variables: Meses, cont") & "<BR><BR>"
        Response.Write("Meses: " & Meses) & "<BR>"
        Response.Write("Contador: 1") & "<BR><BR>"
  For cont = 1 To Meses
    Response.Write("Mes número " & cont) & "<BR>"
  Next
%>
</BODY>
</HTML>
```

[A8]

```
<%@LANGUAGE="VBSCRIPT"%>
<%Option Explicit%>
<%
  Dim matriz(3), contador
  matriz(0) = "Edgar"
  matriz(1) = "Gisela"
  matriz(2) = "Luz Maria"
  matriz(3) = "Lizbeth"
%>
<HTML>
<HEAD>
  <TITLE>EJEMPLO DEL BUCLE FOR...EACH...NEXT</TITLE>
</HEAD>
<BODY>
<%
        Response.Write("EJEMPLO DEL USO DEL BUCLE FOR...EACH...NEXT") & >"<BR><BR>"
        Response.Write("Variables: matriz, contador") & "<BR><BR>"
        Response.Write("Matriz: " & matriz(0)&","&matriz(1)&","&matriz(2)&","&matriz(3)) & "<BR>"
        Response.Write("Contador: No es necesario inicializarlo") & "<BR><BR>"
  For Each contador In matriz
    Response.Write("Nombre: " & contador) & "<BR>"
  Next
```

```
%>
</BODY>
</HTML>
```

[A9]

```
<%@LANGUAGE="VBSCRIPT"%>
<%Option Explicit%>
<%
   Dim condicion, contador
   contador = 1
   condicion = 15
%>
<HTML>
<HEAD>
  <TITLE>EJEMPLO DEL BUCLE DO...WHILE...LOOP</TITLE>
</HEAD>
<BODY>
<%
            Response.Write("EJEMPLO DEL USO DEL BUCLE DO...WHILE...LOOP") & "<BR><BR>"
            Response.Write("Variables: condicion, contador") & "<BR><BR>"
            Response.Write("Condicion: " & condicion) & "<BR>"
            Response.Write("Contador: " & contador) & "<BR><BR>"
   Do While contador <= condicion
     Response.Write("Numero de Iteración: " & contador) & "<BR>"
     contador = contador + 1
   Loop
%>
</BODY>
</HTML>
```

[A10]

```
<%@LANGUAGE="VBSCRIPT"%>
<%Option Explicit%>
<%
   Dim condicion, meses
   meses = 1
   condicion = 12
%>
<HTML>
<HEAD>
  <TITLE>EJEMPLO DEL BUCLE DO...UNTIL...LOOP</TITLE>
</HEAD>
<BODY>
<%
            Response.Write("EJEMPLO DEL USO DEL BUCLE DO...UNTIL...LOOP") & "<BR><BR>"
            Response.Write("Variables: condicion, meses") & "<BR><BR>"
            Response.Write("Condicion: " & condicion) & "<BR>"
            Response.Write("Contador: " & meses) & "<BR><BR>"
   Do Until meses > condicion
     Response.Write("Estamos en el Mes Número: " & meses) & "<br>"
     meses = meses + 1
   Loop
%>
</BODY>
</HTML>
```

[A11]

```
<%@LANGUAGE="VBSCRIPT"%>
<%Option Explicit%>
```

```
<%
   Dim condición, contador
   contador = 1979
   condicion = 2005
%>
<HTML>
<HEAD>
  <TITLE>EJEMPLO DEL BUCLE WHILE...WEND</TITLE>
</HEAD>
<BODY>
<%
          Response.Write("EJEMPLO DEL USO DEL BUCLE WHILE...WEND") & "<BR>"
          Response.Write("Variables: condición, contador") & "<BR>"
          Response.Write("condición: " & condición) & "<BR>"
          Response.Write("Contador: " & contador) & "<BR>"
   While contador <= condición
            If contador = condición Then
                    Response.Write("Año en el que estoy viviendo actualmente: " & contador) & "<BR>"
            Else
                    Response.Write("Años en los que he vivido: " & contador) & "<BR>"
            End If
    contador = contador + 1
   Wend
%>
</BODY>
</HTML>
```

[A12]

```
<%@LANGUAGE="VBSCRIPT"%>
<%Option Explicit%>
<%
          Dim MiNombre, MiPais, MiEdad, MiApellido, i
          Dim FuncionLen, FuncionSplit, FuncionMid, FuncionTrim, FuncionInstr, FuncionReplace
          MiNombre = "Edgar Guerrero"
          MiPais = "Mexico!Brasil"
          MiEdad = " 26 "
          MiApellido = "Guerrero"
          FuncionLen = Len (MiNombre)
          FuncionSplit = Split (MiPais,"!")
          FuncionMid = Mid (MiNombre,7,14)
          FuncionTrim = Trim (MiEdad)
          FuncionInstr = Instr (MiNombre,MiApellido)
          FuncionReplace = Replace (MiPais,"!","@")
%>
<HTML>
<HEAD>
  <TITLE>MANEJO DE FUNCIONES SOBRE CADENAS</TITLE>
</HEAD>
<BODY>
<%
Response.Write("MANEJO DE FUNCIONES SOBRE CADENAS") & "<BR>"
Response.Write("Variables: MiNombre, MiPais, MiEdad, MiApellido") & "<BR>"
Response.Write("Nombre: " & MiNombre) & "<BR>"
Response.Write("Pais: " & MiPais) & "<BR>"
Response.Write("Edad: " & MiEdad) & "<BR>"
Response.Write("Apellido: " & MiApellido) & "<BR><BR>"
Response.Write("Ejemplo de la Función Len:") & "<BR>"
Response.Write("Tamaño de la cadena MiNombre: " & FuncionLen) & "<BR><BR>"
Response.Write("Ejemplo de la Función Split y de la función Ubound dentro de un bucle For:") & "<BR>"
For i = 0 To Ubound(FuncionSplit)
Response.Write("Muestra el contenido de la cadena MiPais ya subdividida: " & FuncionSplit(i)) & "<BR>"
Next
Response.Write("<BR>")
```

```
Response.Write("Ejemplo de la Función Mid:") & "<BR>"
Response.Write("De la cadena MiNombre muestra la palabra Guerrero: " & FuncionMid) & "<BR><BR>"
Response.Write("Ejemplo de la Función Trim:") & "<BR>"
Response.Write("De la cadena MiEdad quita los espacios al inicio y fin: " & FuncionTrim)&"<BR><BR>"
Response.Write("Ejemplo de la Función Instr:") & "<BR>"
Response.Write("En MiNombre busca MiApellido y da la posición: " & FuncionInstr) & "<BR><BR>"
Response.Write("Ejemplo de la Función Replace:") & "<BR>"
Response.Write("En MiPais cambia el carácter ! por el carácter @: " & FuncionReplace) & "<BR><BR>"
%>
</BODY>
</HTML>
```

[A13]

```
<%@LANGUAGE="VBSCRIPT"%>
<%Option Explicit%>
<%
Dim ValorDevuelto
Sub DivisionSub()
        Dim resultado, valor1, valor2
        valor1 = 2027
        valor2 = 23
        resultado = valor1 / valor2
        Response.Write("<B>EJEMPLO DEL PROCEDIMIENTO SUB</B>")&"<BR><BR>"
        Response.Write("<LI>Variables: resultado, valor1, valor2")&"<BR>"
        Response.Write("    valor1: " & valor1)&"<BR>"
        Response.Write("    valor2: " & valor2)&"<BR><BR>"
        Response.Write("<LI>Se realiza la siguiente operación:")&"<BR>"
        Response.Write("    resultado = valor1 / valor2")&"<BR><BR>"
        Response.Write "<LI>El resultado de la división es: " & resultado
End Sub
Function DivisionFunction()
        Dim resultado, valor1, valor2
        valor1 = 2027
        valor2 = 23
        resultado = valor1 / valor2
        DivisionFunction = resultado
End Function
ValorDevuelto = DivisionFunction()
%>
<html>
<head>
  <TITLE>UTILIZACION DE LOS PROCEDIMIENTOS SUB Y FUNCTION</TITLE>
</head>
<body>
<center>
<table border="2">
<tr>
<td>
<%Call DivisionSub()%>
</td>
</tr>
</table>
<br>
<table border="2">
<tr>
<td>
<font face="arial" size="2" color="black"><B>EJEMPLO DEL PROCEDIMIENTO FUNCTION</b></font>
<br><br>
<font face="arial" size="2" color="black"><li>Variables: resultado, valor1, valor2</font>
<br><br>
<font face="arial" size="2" color="black"><li>Se realiza la siguiente operación:</font>
<br>
<font face="arial" size="2" color="black">    resultado=valor1/ valor2</font>
```

```
<br><br>
<font face="arial" size="2" color="black"><li>El resultado de la división es: <%=ValorDevuelto%></font>
</td>
</tr>
</table>
</center>
</body>
</html>
```

[A14]

```
<%@LANGUAGE="VBSCRIPT"%>
<%Option Explicit%>
<%Dim resultado%>
<!-- #include file="ejemplo_libreria.asp" -->
<html>
<head>
  <title>Ejemplo de ASP</title>
</head>
<body>
<%Call CabeceraPagina()
 resultado = CuerpoPagina()%>
<br>
<center>
<table border="2">
<tr>
<td align="middle">
<font face="arial" size="4" color="blue"><b>EJEMPLO DE LIBRERIAS</b></font>
</td>
</tr>
<tr>
<td>
<font face="arial" size="2" color="blue"><li>Esta página ejecuta Procedimientos que se encuentran declarados en otro archivo y que</font><br>
<font face="arial" size="2" color="blue">&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;se muestran en el encabezado y pie de esta pagina, así como la fecha del día hoy.</font>
</td>
</tr>
<tr>
<td align="left">
<font face="arial" size="2" color="blue"><li>Hoy estamos a:&amp;nbsp;<%=resultado%></font>
</td>
</tr>
</table>
</center>
<br><br>
<%Call PiePagina()%>
</body>
</html>
```

[A15]

```
<%@LANGUAGE="VBSCRIPT"%>
<%Option Explicit%>
<html>
<head>
<title>uso de formularios</title>
</head>
<body>
<form action="mostrar_formulario.asp" id="form1" name="form1" method="post">
<center>
<tr>
<td>
```

```
<font face="arial" color="black" size="4"><b>Ejemplo del uso de formularios</b></font>
</td>
</tr>
<br><br><br>
<table width="50%" align="middle" border="2">
<tr>
<td width="50%" align="left">
<font face="arial" color="black" size="2">Nombre :</font>
</td>
<td width="50%" align="left">
<input id="txtnombre" name="txtnombre" >
</td>
</tr>
<tr>
<td width="50%" align="left">
<font face="arial" color="black" size="2">Edad :</font>
</td>
<td width="50%" align="left">
<input id="txtedad" name="txtedad" size="2" maxlength="2">
</td>
</tr>
<tr>
<td width="50%" align="left">
<font face="arial" color="black" size="2">Sexo :</font>
</td>
<td width="50%" align="left">
<font face="arial" color="black" size="2">M</font>
<input type="radio" id="rbdsexo" name="rbdsexo" value="M">
<font face="arial" color="black" size="2">F</font>
<input type="radio" id="rbdsexo" name="rbdsexo" value="F">
</td>
</tr>
<tr>
<td width="50%" align="left">
<font face="arial" color="black" size="2"Selecciona un País :</font>
</td>
<td width="50%" align="left">
<select id="cmbpais" name="cmbpais">
<option value="0" selected> - selecciona una opción - </option>
<option value="1">México</option>
<option value="2">Canada</option>
<option value="3">Cuba</option>
<option value="4">Alemania</option>
<option value="5">Francia</option>
<option value="6">Brasil</option>
</select>
</td>
</tr>
</table>
<br>
<input type="submit" id="btnaceptar" name="btnaceptar" value="Enviar Información">
</center>
</form>
</body>
</html>
```

[A16]

```
<%@LANGUAGE="VBSCRIPT"%>
<%Option Explicit%>
<html>
<head>
<title>Uso de Formularios</title>
</head>
```

```
<body>
<form action="mostrar_formulario.asp" id="form1" name="form1" method="post" onsubmit="return
Validar_Datos()">
<center>
<tr>
<td>
<font face="arial" color="black" size="4"><b>Ejemplo del uso de Formularios</b></font>
</td>
</tr>
<br><br><br>
<table width="50%" align="middle" border="2">
<tr>
<td width="50%" align="left">
<font face="arial" color="black" size="2">Nombre :</font>
</td>
<td width="50%" align="left">
<input id="txtnombre" name="txtnombre" >
</td>
</tr>
<tr>
<td width="50%" align="left">
<font face="arial" color="black" size="2">Edad :</font>
</td>
<td width="50%" align="left">
<input id="txtedad" name="txtedad" size="2" maxlength="2" onblur="Valida_Numeros(this)">
</td>
</tr>
<tr>
<td width="50%" align="left">
<font face="arial" color="black" size="2">Sexo :</font>
</td>
<td width="50%" align="left">
<font face="arial" color="black" size="2">M</font>
<input type="radio" id="rbdsexo" name="rbdsexo" value="M">
<font face="arial" color="black" size="2">F</font>
<input type="radio" id="rbdsexo" name="rbdsexo" value="F">
</td>
</tr>
<tr>
<td width="50%" align="left">
<font face="arial" color="black" size="2"Selecciona un País :</font>
</td>
<td width="50%" align="left">
<select id="cmbpais" name="cmbpais">
<option value="0" selected> - selecciona una opción - </option>
<option value="1">México</option>
<option value="2">Canada</option>
<option value="3">Cuba</option>
<option value="4">Alemania</option>
<option value="5">Francia</option>
<option value="6">Brasil</option>
</select>
</td>
</tr>
</table>
<br>
<input type="submit" id="btnaceptar" name="btnaceptar" value="Enviar Información">
</center>
</form>
</body>
</html>

<script language="JavaScript">
        function Validar_Datos()
        {
```

```
		if(document.form1.txtNombre.value == "")
		{
			alert("Debe indicar su nombre completo.")
			document.form1.txtNombre.focus()
			return(false)
		}
		if(document.form1.txtEdad.value == "")
		{
			alert("Debe indicar su edad.")
			document.form1.txtEdad.focus()
			return(false)
		}

		var cadena = 0
		for(i=0;i<document.form1.elements.length;i++)
		{
			if(document.form1.elements[i].type == "radio")
			{
				if(document.form1.elements[i].checked == true)
				{
					cadena = 1
				}
			}
		}
		if(cadena == 0)
		{
			alert("Debe indicar su sexo.")
			return(false)
		}
		if(document.form1.cmbPais.value == "0")
		{
			alert("Debe seleccionar su país de origen.")
			document.form1.cmbPais.focus()
			return(false)
		}
		if(confirm("¿Todos los datos son correctos?") == true)
		{
			return(true)
		}
		else
		{
			return(false)
		}
}

function Valida_Numeros(dato)
{
  var cadenaOK = "0123456789";
  var cadenaStr = dato.value;
  var Valido = true;
  for(i=0;i<cadenaStr.length;i++)
  {
    ch = cadenaStr.charAt(i);
    for(j=0;j<cadenaOK.length;j++)
      if(ch == cadenaOK.charAt(j))
        break;
      if(j == cadenaOK.length)
      {
          Valido = false;
          break;
      }
  }
  if (!Valido)
  {
		alert("Escriba sólo dígitos en este campo.");
```

```
                    dato.value="";
                    dato.focus();
                    return (false);
            }
           }
</script>
```

[A17]

```
<%@LANGUAGE="VBSCRIPT"%>
<%Option Explicit%>
<%
        Application("Nombre") = "Edgar Guerrero"
        Application("Apellidos") = "Guerrero González"
        Application("Edad") = 26
        Application("n_visitas") = Application("n_visitas") + 1
%>
<html>
<head>
  <title>Uso del Objeto Application</title>
</head>
<body>
<form action="ejemplo_application.asp" id="form1" name="form1" method="post">
 <center>
  <tr>
   <td>
    <font face="arial" color="black" size="4"><b>Ejemplo del Uso del Objeto Application</b></font>
   </td>
  </tr>
  <br><br>
  <tr>
   <td>
    <font face="arial" color="black" size="2"><b>Hola, Bienvenido a mi Página Web</b></font>
   </td>
  </tr>
  <tr>
   <td>
    <font face="arial" color="black" size="2"><b>eres el visitante número:
<%=application("n_visitas")%></b></font>
   </td>
  </tr>
 <br><br><br>
 <table width="50%" align="middle" border="2">
  <tr>
   <td width="50%" align="left">
    <font face="arial" color="black" size="2">Nombre(s):</font>
   </td>
   <td width="50%" align="left">
    <input id="txtnombre" name="txtnombre" value="<%=Application("Nombre")%>">
   </td>
  </tr>
  <tr>
   <td width="50%" align="left">
    <font face="arial" color="black" size="2">Apellidos:</font>
   </td>
   <td width="50%" align="left">
    <input id="txtapellidos" name="txtapellidos" value="<%=Application("Apellidos")%>">
   </td>
  </tr>
  <tr>
   <td width="50%" align="left">
    <font face="arial" color="black" size="2">Edad:</font>
   </td>
   <td width="50%" align="left">
```

```
    <input id="txtedad" name="txtedad" size="2" maxlength="2" value="<%=Application("Edad")%>">
   </td>
  </table>
  <br>
  <input type="submit" id="btnaceptar" name="btnaceptar" value="Enviar Información">
 </center>
</form>
</body>
</html>
```

[A18]

```
<%@LANGUAGE="VBSCRIPT"%>
<%Option Explicit%>
<%
        If Request.QueryString("envio_datos") = "SI" Then
                Dim nombre, apellido_p, apellido_m, edad, sexo, estado
                nombre = Request.QueryString("txtNombre")
                apellido_p = Request.QueryString("txtApePat")
                apellido_m = Request.QueryString("txtApeMat")
                edad = Request.QueryString("txtEdad")
                If Request.QueryString("rbtSexo") = "F" Then
                        sexo = "Mujer"
                ElseIf Request.QueryString("rbtSexo") = "M" Then
                        sexo = "Hombre"
                Else
                        sexo = "No Definido"
                End If
                estado = Request.QueryString("cmbEstado")
        End If
%>
<html>
<body>
<head>
<title>Ejemplo del Uso del Objeto Request</title>
</head>
<form action="objeto_request.asp" id="form1" name="form1" method="get" onsubmit="return Aceptar()">
<center>
<tr>
<td width="100%" align="left">
<font face="arial" color="black" size="4"><b>EJEMPLO DEL OBJETO REQUEST<b></font>
</td>
</tr>
<br>
<%If Request.QueryString("envio_datos") = "SI" Then%>
<tr>
<td width="100%" align="left">
<font face="arial" color="black" size="2"><b>COLECCION QUERYSTRING<b></font>
</td>
</tr>
<br><br>
<table border="2" width="60%">
<tr>
        <td width="35%" align="left">
                <font face="arial" color="black" size="2">Nombre(s):</font>
        </td>
        <td width="60%" align="left">
                <font face="arial" color="black" size="2"> <%=nombre%></font>
        </td>
</tr>
<tr>
        <td width="35%" align="left">
                <font face="arial" color="black" size="2">Apellido Paterno:</font>
        </td>
```

```
		<td width="60%" align="left">
			<font face="arial" color="black" size="2"> <%=apellido_p%></font>
		</td>
</tr>
<tr>
		<td width="35%" align="left">
			<font face="arial" color="black" size="2">Apellido Materno:</font>
		</td>
		<td width="60%" align="left">
			<font face="arial" color="black" size="2"> <%=apellido_m%></font>
		</td>
</tr>
<tr>
		<td width="35%" align="left">
			<font face="arial" color="black" size="2">Edad:</font>
		</td>
		<td width="60%" align="left">
			<font face="arial" color="black" size="2"> <%=edad%> Años</font>
		</td>
</tr>
<tr>
		<td width="35%" align="left">
			<font face="arial" color="black" size="2">Sexo:</font>
		</td>
		<td width="60%" align="left">
			<font face="arial" color="black" size="2"> <%=sexo%></font>
		</td>
</tr>
<tr>
		<td width="35%" align="left">
			<font face="arial" color="black" size="2">Estado:</font>
		</td>
		<td width="60%" align="left">
			<font face="arial" color="black" size="2"> <%=estado%></font>
		</td>
</tr>
</table>
<br>
<input type="button" id="btnRegresar" name="btnRegresar" value="Regresar" onclick="Regresar()">
<%Else%>
<table border="2" width="60%">
<tr>
		<td width="35%" align="left">
			<font face="arial" color="black" size="2">Nombre(s):</font>
		</td>
		<td width="60%" align="left">
			<input type="text" id="txtNombre" name="txtNombre" size="25" maxlength="20">
		</td>
</tr>
<tr>
		<td width="35%" align="left">
			<font face="arial" color="black" size="2">Apellido Paterno:</font>
		</td>
		<td width="60%" align="left">
			<input type="text" id="txtApePat" name="txtApePat" size="25" maxlength="20">
		</td>
</tr>
<tr>
		<td width="35%" align="left">
			<font face="arial" color="black" size="2">Apellido Materno:</font>
		</td>
		<td width="60%" align="left">
			<input type="text" id="txtApeMat" name="txtApeMat" size="25" maxlength="20">
		</td>
</tr>
```

```
<tr>
        <td width="35%" align="left">
                <font face="arial" color="black" size="2">Edad:</font>
        </td>
<td width="60%" align="left">
<input type="text" id="txtEdad" name="txtEdad" size="2" maxlength="2"> <font face="arial" color="black"
size="2">Años</font>
</td>
</tr>
<tr>
        <td width="35%" align="left">
                <font face="arial" color="black" size="2">Sexo:</font>
        </td>
        <td width="60%" align="left">
                <font face="arial" color="black" size="2">Mujer</font>
                <input type="radio" id="rbtSexo" name="rbtSexo" value="F">
                <font face="arial" color="black" size="2">Hombre</font>
                <input type="radio" id="rbtSexo" name="rbtSexo" value="M">
        </td>
</tr>
<tr>
        <td width="35%" align="left">
                <font face="arial" color="black" size="2">Selecciona un Estado:</font>
        </td>
        <td width="60%" align="left">
                <select id="cmbEstado" name="cmbEstado">
                <option value="0"> - Selecciona el Estado donde vives - </option>
                <option value="Aguascalientes">1.- Aguascalientes</option>
                <option value="Baja California Norte">2.- Baja California Norte</option>
                <option value="Baja California Sur">3.- Baja California Sur</option>
                <option value="Campeche">4.- Campeche</option>
                <option value="Coahuila">5.- Coahuila</option>
                <option value="Colima">6.- Colima</option>
                <option value="Chiapas">7.- Chiapas</option>
                <option value="Chihuahua">8.- Chihuahua</option>
                <option value="Distrito Federal">9.- Distrito Federal</option>
                <option value="Durango">10.- Durango</option>
                <option value="Guanajuato">11.- Guanajuato</option>
                <option value="Guerrero">12.- Guerrero</option>
                <option value="Hidalgo">13.- Hidalgo</option>
                <option value="Jalisco">14.- Jalisco</option>
                <option value="Mexico">15.- Mexico</option>
                <option value="Michoacan">16.- Michoacan</option>
                <option value="Morelos">17.- Morelos</option>
                <option value="Nayarit">18.- Nayarit</option>
                <option value="Nuevo Leon">19.- Nuevo Leon</option>
                <option value="Oaxaca">20.- Oaxaca</option>
                <option value="Puebla">21.- Puebla</option>
                <option value="Queretaro">22.- Queretaro</option>
                <option value="Quintana Roo">23.- Quintana Roo</option>
                <option value="San Luis Potosi">24.- San Luis Potosi</option>
                <option value="Sinaloa">25.- Sinaloa</option>
                <option value="Sonora">26.- Sonora</option>
                <option value="Tabasco">27.- Tabasco</option>
                <option value="Tamaulipas">28.- Tamaulipas</option>
                <option value="Tlaxcala">29.- Tlaxcala</option>
                <option value="Veracruz">30.- Veracruz</option>
                <option value="Yucatan">31.- Yucatan</option>
                <option value="Zacatecas">32.- Zacatecas</option>
                </select>
        </td>
        </tr>
</table>
<br>
<input type="submit" id="btnAceptar" name="btnAceptar" value="Aceptar">
```

```
<input type="hidden" id="envio_datos" name="envio_datos">
<%End If%>
</center>
</form>
</body>
</html>
<script language="JavaScript">
	function Aceptar()
	{
		if(document.form1.txtNombre.value == "")
		{
			alert("Debes indicar tu(s) Nombre(s).")
			document.form1.txtNombre.focus()
			return(false)
		}
		if(document.form1.txtApePat.value == "")
		{
			alert("Debes indicar tu Apellido Paterno.")
			document.form1.txtApePat.focus()
			return(false)
		}
		if(document.form1.txtApeMat.value == "")
		{
			alert("Debes indicar tu Apellido Materno.")
			document.form1.txtApeMat.focus()
			return(false)
		}
		if(document.form1.txtEdad.value == "")
		{
			alert("Debes indicar tu Edad.")
			document.form1.txtEdad.focus()
			return(false)
		}
		var cadena = 0
		for(i=0;i<document.form1.elements.length;i++)
		{
			if(document.form1.elements[i].type == "radio")
			{
				if(document.form1.elements[i].checked == true)
				{
					cadena = 1
				}
			}
		}
		if(cadena == 0)
		{
			alert("Debes indicar tu Sexo.")
			return(false)
		}
		if(document.form1.cmbEstado.value == "0")
		{
			alert("Debes seleccionar un Estado.")
			document.form1.cmbEstado.focus()
			return(false)
		}
		document.form1.envio_datos.value = "SI"
	}

	function Regresar()
	{
		location.href = "objeto_request.asp"
	}
</script>
```

[A19]

```
<%@LANGUAGE="VBSCRIPT"%>
<%Option Explicit%>
<%
	If Request.Form("envio_datos") = "SI" Then
		Dim nombre, apellido_p, apellido_m, edad, sexo, estado
		nombre = Request.Form("txtNombre")
		apellido_p = Request.Form("txtApePat")
		apellido_m = Request.Form("txtApeMat")
		edad = Request.Form("txtEdad")
		If Request.Form("rbtSexo") = "F" Then
			sexo = "Mujer"
		ElseIf Request.Form("rbtSexo") = "M" Then
			sexo = "Hombre"
		Else
			sexo = "No Definido"
		End If
		estado = Request.Form("cmbEstado")
	End If
%>
<html>
<body>
<head>
<title>Ejemplo del Uso del Objeto Request</title>
</head>
<form action="objeto_request_form.asp" id="form1" name="form1" method="post" onsubmit="return Aceptar()">
<center>
<tr>
<td width="100%" align="left">
<font face="arial" color="black" size="4"><b>EJEMPLO DEL OBJETO REQUEST<b></font>
</td>
</tr>
<br>
<%If Request.Form("envio_datos") = "SI" Then%>
<tr>
<td width="100%" align="left">
<font face="arial" color="black" size="2"><b>COLECCION FORM<b></font>
</td>
</tr>
<br><br>
<table border="2" width="60%">
<tr>
	<td width="35%" align="left">
		<font face="arial" color="black" size="2">Nombre(s):</font>
	</td>
	<td width="60%" align="left">
		<font face="arial" color="black" size="2"> <%=nombre%></font>
	</td>
</tr>
<tr>
	<td width="35%" align="left">
		<font face="arial" color="black" size="2">Apellido Paterno:</font>
	</td>
	<td width="60%" align="left">
		<font face="arial" color="black" size="2"> <%=apellido_p%></font>
	</td>
</tr>
<tr>
	<td width="35%" align="left">
		<font face="arial" color="black" size="2">Apellido Materno:</font>
	</td>
	<td width="60%" align="left">
		<font face="arial" color="black" size="2"> <%=apellido_m%></font>
```

```
		</td>
</tr>
<tr>
		<td width="35%" align="left">
			<font face="arial" color="black" size="2">Edad:</font>
		</td>
		<td width="60%" align="left">
			<font face="arial" color="black" size="2"> <%=edad%> Años</font>
		</td>
</tr>
<tr>
		<td width="35%" align="left">
			<font face="arial" color="black" size="2">Sexo:</font>
		</td>
		<td width="60%" align="left">
			<font face="arial" color="black" size="2"> <%=sexo%></font>
		</td>
</tr>
<tr>
		<td width="35%" align="left">
			<font face="arial" color="black" size="2">Estado:</font>
		</td>
		<td width="60%" align="left">
			<font face="arial" color="black" size="2"> <%=estado%></font>
		</td>
</tr>
</table>
<br>
<input type="button" id="btnRegresar" name="btnRegresar" value="Regresar" onclick="Regresar()">
<%Else%>
<table border="2" width="60%">
<tr>
		<td width="35%" align="left">
			<font face="arial" color="black" size="2">Nombre(s):</font>
		</td>
		<td width="60%" align="left">
			<input type="text" id="txtNombre" name="txtNombre" size="25" maxlength="20">
		</td>
</tr>
<tr>
		<td width="35%" align="left">
			<font face="arial" color="black" size="2">Apellido Paterno:</font>
		</td>
		<td width="60%" align="left">
			<input type="text" id="txtApePat" name="txtApePat" size="25" maxlength="20">
		</td>
</tr>
<tr>
		<td width="35%" align="left">
			<font face="arial" color="black" size="2">Apellido Materno:</font>
		</td>
		<td width="60%" align="left">
			<input type="text" id="txtApeMat" name="txtApeMat" size="25" maxlength="20">
		</td>
</tr>
<tr>
		<td width="35%" align="left">
			<font face="arial" color="black" size="2">Edad:</font>
		</td>
<td width="60%" align="left">
<input type="text" id="txtEdad" name="txtEdad" size="2" maxlength="2"> <font face="arial" color="black"
size="2">Años</font>
</td>
</tr>
<tr>
```

```
		<td width="35%" align="left">
			<font face="arial" color="black" size="2">Sexo:</font>
		</td>
		<td width="60%" align="left">
			<font face="arial" color="black" size="2">Mujer</font>
			<input type="radio" id="rbtSexo" name="rbtSexo" value="F">
			<font face="arial" color="black" size="2">Hombre</font>
			<input type="radio" id="rbtSexo" name="rbtSexo" value="M">
		</td>
</tr>
<tr>
		<td width="35%" align="left">
			<font face="arial" color="black" size="2">Selecciona un Estado:</font>
		</td>
		<td width="60%" align="left">
			<select id="cmbEstado" name="cmbEstado">
			<option value="0"> - Selecciona el Estado donde vives - </option>
			<option value="Aguascalientes">1.- Aguascalientes</option>
			<option value="Baja California Norte">2.- Baja California Norte</option>
			<option value="Baja California Sur">3.- Baja California Sur</option>
			<option value="Campeche">4.- Campeche</option>
			<option value="Coahuila">5.- Coahuila</option>
			<option value="Colima">6.- Colima</option>
			<option value="Chiapas">7.- Chiapas</option>
			<option value="Chihuahua">8.- Chihuahua</option>
			<option value="Distrito Federal">9.- Distrito Federal</option>
			<option value="Durango">10.- Durango</option>
			<option value="Guanajuato">11.- Guanajuato</option>
			<option value="Guerrero">12.- Guerrero</option>
			<option value="Hidalgo">13.- Hidalgo</option>
			<option value="Jalisco">14.- Jalisco</option>
			<option value="Mexico">15.- México</option>
			<option value="Michoacan">16.- Michoacan</option>
			<option value="Morelos">17.- Morelos</option>
			<option value="Nayarit">18.- Nayarit</option>
			<option value="Nuevo Leon">19.- Nuevo Leon</option>
			<option value="Oaxaca">20.- Oaxaca</option>
			<option value="Puebla">21.- Puebla</option>
			<option value="Queretaro">22.- Querétaro</option>
			<option value="Quintana Roo">23.- Quintana Roo</option>
			<option value="San Luis Potosi">24.- San Luis Potosí</option>
			<option value="Sinaloa">25.- Sinaloa</option>
			<option value="Sonora">26.- Sonora</option>
			<option value="Tabasco">27.- Tabasco</option>
			<option value="Tamaulipas">28.- Tamaulipas</option>
			<option value="Tlaxcala">29.- Tlaxcala</option>
			<option value="Veracruz">30.- Veracruz</option>
			<option value="Yucatan">31.- Yucatan</option>
			<option value="Zacatecas">32.- Zacatecas</option>
</select>
		</td>
		</tr>
</table>
<br>
<input type="submit" id="btnAceptar" name="btnAceptar" value="Aceptar">
<input type="hidden" id="envio_datos" name="envio_datos">
<%End If%>
</center>
</form>
</body>
</html>
<script language="JavaScript">
		function Aceptar()
		{
			if(document.form1.txtNombre.value == "")
```

```
		{
			alert("Debes indicar tu(s) Nombre(s).")
			document.form1.txtNombre.focus()
			return(false)
		}
		if(document.form1.txtApePat.value == "")
		{
			alert("Debes indicar tu Apellido Paterno.")
			document.form1.txtApePat.focus()
			return(false)
		}
		if(document.form1.txtApeMat.value == "")
		{
			alert("Debes indicar tu Apellido Materno.")
			document.form1.txtApeMat.focus()
			return(false)
		}
		if(document.form1.txtEdad.value == "")
		{
			alert("Debes indicar tu Edad.")
			document.form1.txtEdad.focus()
			return(false)
		}
		var cadena = 0
		for(i=0;i<document.form1.elements.length;i++)
		{
			if(document.form1.elements[i].type == "radio")
			{
				if(document.form1.elements[i].checked == true)
				{
					cadena = 1
				}
			}
		}
		if(cadena == 0)
		{
			alert("Debes indicar tu Sexo.")
			return(false)
		}
		if(document.form1.cmbEstado.value == "0")
		{
			alert("Debes seleccionar un Estado.")
			document.form1.cmbEstado.focus()
			return(false)
		}
		document.form1.envio_datos.value = "SI"
	}

	function Regresar()
	{
		location.href = "objeto_request_form.asp"
	}
</script>
```

[A20]

```
<%@LANGUAGE="VBSCRIPT"%>
<%Option Explicit%>
<%
If Request.Form.Count = 0 Then
  If Request.Cookies("registro").HasKeys = 0 Then%>
	<html>
	<head>
	<title>Ejemplo del Uso del Objeto Request</title>
```

```
            </head>
            <body>
            No se encuentra registrado en nuestra Web,
            por favor complete el formulario siguiente:
            <form action="objeto_request_cookies.asp" method="post" id=form1 name=form1>
            <table>
            <tr>
            <td>
            <input type="text" name="nombre" id="nombre"size="20">
            </td>
            <td>
            <input type="Submit" value="Crear Cookie!" id="Aceptar" name="Aceptar">
            </td>
            </tr>
            </table>
            </form>
            </body>
            </html>
<%
 Else
            Dim iCont
            iCont = Request.Cookies("registro")("entradas")
            Response.Cookies("registro")("entradas") = iCont + 1
            Response.Write "Bienvenido: " & Request.Cookies("registro")("nombre") &"<br>"
            Response.Write "Has entrado " & iCont & " veces"
 End If
Else
 Response.Cookies("registro")("nombre") = Request.Form("nombre")
 Response.Cookies("registro")("entradas") = 1
 Response.Write "Gracias por Registrarse, "
 Response.Write Request.Cookies("registro")("nombre")&"<br><br>"%>
<input type="button" value="Regresar" id="Regresar" name="Regresar" onclick="Regresar()">
<%End If%>
<script language="javascript">
            Function Regresar()
            {
                        Location.href = "objeto_request_cookies.asp"
            }
</script>
```

[A21]

```
<%@LANGUAGE=VBSCRIPT%>
<%Option Explicit%>
<%
'Ejemplo para conectarnos a base de datos.
Dim Conexion, Comando, RecordSQL, usuario,direccion,sexo
'Creamos el objeto Connection.
Set Conexion = CreateObject("ADODB.Connection")
'Ejecutamos el DSN para conectarnos a la base de datos.
Conexion.Open = DSN=Sistema_Personal;UserID=Personal;Password=Personal;Database=SistemaPersonal"
'Creamos el objeto Command.
Set Comando = CreateObject("ADODB.Command")
'Creamos el objeto RecordSet.
Set RecordSQL = CreateObject("ADODB.RecordSet")
'Utilizamos la propiedad CommandText para ejecutar la sentencia de SQL.
Comando.CommandText = "SELECT
nombre,apellido_paterno,apellido_materno,calle,colonia,municipio,fecha_nacimiento,sexo,estado_civil FROM
usuarios"
'Le asignamos la conexión al objeto Command.
Comando.ActiveConnection = Conexion
'Ejecutamos el objeto Command y lo guardamos en el objeto RecordSet.
Set RecordSQL = Comando.Execute
%>
```

```
<html>
<body>
<form>
<table border="2">
<tr>
<td width="25%" align="middle" bgcolor="red">
<font face="arial" size="2" color="white"><b>NOMBRE</b></font>
</td>
<td width="30%" align="middle" bgcolor="red">
<font face="arial" size="2" color="white"><b>DIRECCIÓN</b></font>
</td>
<td width="10%" align="middle" bgcolor="red">
<font face="arial" size="2" color="white"><b>FECHA DE NACIMIENTO</b></font>
</td>
<td width="10%" align="middle" bgcolor="red">
<font face="arial" size="2" color="white"><b>SEXO</b></font>
</td>
<td width="10%" align="middle" bgcolor="red">
<font face="arial" size="2" color="white"><b>ESTADO CIVIL</b></font>
</td>
</tr>
<%
'Recorremos los registros obtenidos para pintarlos.
While Not RecordSQL.EOF

usuario = RecordSQL("nombre")&" "&RecordSQL("apellido_paterno")&" "&RecordSQL("apellido_materno")
direccion = "<b>Calle:</b> "&RecordSQL("calle")&", <b>Colonia:</b> "&RecordSQL("colonia")&",
<b>Municipio:</b> "&RecordSQL("municipio")
If RecordSQL("sexo") = "M" Then
        sexo = "Hombre"
Else
        sexo = "Mujer"
End If
%>
<tr>
<td align="left">
<font face="arial" size="2" color="black"><%=usuario%></font>
</td>
<td align="left">
<font face="arial" size="2" color="black"><%=direccion%></font>
</td>
<td align="middle">
<font face="arial" size="2" color="black"><%=RecordSQL("fecha_nacimiento")%></font>
</td>
<td align="middle">
<font face="arial" size="2" color="black"><%=sexo%></font>
</td>
<td align="middle">
<font face="arial" size="2" color="black"><%=RecordSQL("estado_civil")%></font>
</td>
<tr>
<%
 'Nos movemos al siguiente registro.
 RecordSQL.MoveNext
Wend%>
</table>
</form>
</body>
</html>
```

[A22]

```
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
```

```
Sub Application_OnStart
'Las configuraciones las buscamos en la BD, las obtenemos e inicializamos las variables de aplicación del sitio.
Dim Conexion, Comando, Consulta
Set Conexion = CreateObject("ADODB.Connection")
Conexion.Open = DSN=Sistema_Personal;UserID=Personal;Password=Personal;Database=SistemaPersonal"
Set Comando = CreateObject("ADODB.Command")
Set Consulta = CreateObject("ADODB.RecordSet")
Comando.CommandText = " SELECT id_var_app, descripcion, valor FROM trc_var_app "
Comando.ActiveConnection = Conexion
Set Consulta = Comando.Execute

'-----------------------------------------------------------------
'GENERACION DE LA VARIABLE DE APLICACIÓN DEL CONSECUTIVO(XX) DEL CLIENTE
Application("uv") = Consulta(2)
Consulta.MoveNext
'-----------------------------------------------------------------
'GENERACION DE VARIABLE DE APLICACION PARA EL NOMBRE DE LA BD DE APOYOS
Application("DB_Apoyo_Acad")= Consulta(2)
Consulta.MoveNext
'-----------------------------------------------------------------
'GENERACION DE LA VARIABLE DE APLICACION PARA NOMBRE O DIRECCIÓN DEL SERVIDOR
Application("server") = Consulta(2)
Consulta.MoveNext
'-----------------------------------------------------------------
'GENERACION DE LA VARIABLE DE APLICACION PARA LA RUTA DESTINO DE LOS ARCHIVOS DE CADA
CHAT
Application("chat_route") = Consulta(2)
Consulta.MoveNext
'-----------------------------------------------------------------
'GENERACION DE LA VARIABLE DE APLICACION PARA VER SI QUIERE APOYO ACADEMICO O NO
Application("apoyo_acad") = Consulta(2)
Consulta.MoveNext
'-----------------------------------------------------------------
'GENERACION DE LA VARIABLE DE APLICACION PARA LA IP DONDE SE ENCONTRARA EL SITIO DE APOYO
ACADEMICO
Application("ip") = Consulta(2)
Consulta.MoveNext
'-----------------------------------------------------------------
'GENERACION DE LA VARIABLE DE APLICACION PARA EL EMAIL DEL CLIENTE
Application("mail") = Consulta(2)
Consulta.MoveNext
'-----------------------------------------------------------------
'SITIO WEB DE TRALCOM INICIAL DE TODO EL SISTEMA
Application("init_web") = Consulta(2)
Consulta.MoveNext
'-----------------------------------------------------------------
'GENERACION DE LA VARIABLE DE APLICACION PARA INDICAR LA RUTA DE
'LAS IMAGENES DE LOS APOYOS ACADEMICOS
Application("ruta_img_apoyos") = Consulta(2)
Consulta.MoveNext
'-----------------------------------------------------------------
'GENERACION DE LA VARIABLE DE APLICACION PARA INDICAR LA RUTA WEB
'DE LOS SITIOS DE APOYO ACADEMICO
Application("ruta_web_apoyos") = Consulta(2)
Consulta.MoveNext
'-----------------------------------------------------------------
'GENERACION DE LA VARIABLE DE APLICACION PARA INDICAR LA LONGITUD
'MINIMA DE LA CONTRASEÑA DE ACCESO AL SISTEMA
Application("long_pwd") = Consulta(2)
Consulta.MoveNext
'-----------------------------------------------------------------
'VARIABLE QUE CONTIENE LA DESCRIPCIÓN DEL EMPLEADO DE ACUERDO AL CLIENTE(ALUMNO,
ESTUDIANTE, ETC.)
Application("empleado") =  Consulta(2)
Consulta.MoveNext
```

```
'-----------------------------------------------------------------------
'VARIABLE QUE INDICA SI EL SITIO SOPORTA EL ENVIO DE MENSAJES INSTANTANEOS(L.C.S)
Application("req_msn") = Consulta(2)
Consulta.MoveNext
'-----------------------------------------------------------------------
SET obj=NOTHING


'-----------------------------------------------------
'GENERACION DE LAS VARIABLES DE APLICACION DE DRM
dim oDBInfo

on error resume next
Set oDBInfo = Server.CreateObject("DRMInfo.DRMDatabaseInfo")
If Err.number = 0 Then
 Application("DBSERVER") = oDBInfo.Server
 Application("DBDATABASE") = oDBInfo.Database
 Application("DBUID") = oDBInfo.UID
 Application("DBPASSWORD") = oDBInfo.Password
 Application("ConnectionString") = oDBInfo.ConnectionString
Set oDBInfo = Nothing
Else
 Response.Write "Error occurred : " & Err.description
End If
'-----------------------------------------------------
Application("ReiniciarSesion") = "http://localhost/tc-emp-xx/"
End Sub

Sub Application_OnEnd
  'Check the rutines for the end of session
End Sub

Sub Session_OnStart
Session.Timeout=200
Session("DBConn_ConnectionString2")="DRIVER=SQLServer;SERVER=Zimapan;APP=trainingcoord_XX;WSID=Z
imapan;DATABASE=TrainingCoord_XX;User Id=training_XX;PASSWORD=tnttc_XX;"
Session("DBConn_ConnectionTimeout2") = 15
Session("DBConn_CommandTimeout2") = 30
Session("DBConn_CursorLocation2") = 3
Session("DBConn_RuntimeUserName2") = ""
Session("DBConn_RuntimePassword2") = ""

End Sub

Sub Session_OnEnd
  'Check the rutines for the end of session
End Sub

</script>
```

[A23]

```
ape_pat = rsig(2)
ape_mat = rsig(3)
nom1 = rsig(4)
nom2 = rsig(5)
email = rsig(6)
<form name="form1" id="form1" method="post"
action="mod_usuarios.asp?division=<%=request("division")%>&usuario=<%=request("usuario")%>&nom_usuari
o=<%=server.URLEncode(request("nom_usuario"))%>">

<center>
<img src="../../images/<%=session.contents("img_cliente")%>banners/badministracion.gif" name="btnAlta"
id="btnAlta" title="Alta de Cursos" WIDTH="462" HEIGHT="39"><br>
```

```
<font face="tahoma" size="2" color="<%=session.contents("color_text")%>"><strong>MODIFICACION DE USUARIOS</strong></font>
</center>
<br>
<center>

<table border="0" width="100%">
<tr>
<td width="25%" align="left">
<font face="tahoma" size="1" color="<%=session.contents("color_text")%>">Usuario:</font></td>
<td width="25%" align="left">
<input type="text" id="txtusu" name="txtusu" maxlength="15" size="15" value="<%=usuario_alfa%>" style="font-size:8pt;font-family:arial,verdana" disabled></td>
<td width="25%" align="right"></td>
<td width="25%" align="left"></td>
</tr>
<tr>
<td width="25%" align="left">
<font face="tahoma" size="1" color="<%=session.contents("color_text")%>"> Apellido Paterno</font></td>
<td width="25%" align="left">
<input type="text" id="txtapepat" name="txtapepat" maxlength="30" size="30" onblur="ConvierteMay(this)" value="<%=ape_pat%>" style="font-size:8pt;font-family:arial,verdana" disabled></td>
<td width="25%" align="right">
<font face="tahoma" size="1" color="<%=session.contents("color_text")%>">Apellido Materno</font></td>
<td width="25%" align="left">
<input type="text" id="txtapemat" name="txtapemat" maxlength="30" size="30" onblur="ConvierteMay(this)" value="<%=ape_mat%>" style="font-size:8pt;font-family:arial,verdana" disabled></td>
</tr>
<tr>
<td width="25%" align="left">
<font face="tahoma" size="1" color="<%=session.contents("color_text")%>">Primer Nombre</font></td>
<td width="25%" align="left">
<input type="text" id="txtnombre1" name="txtnombre1" maxlength="30" size="30" onblur="ConvierteMay(this)" value="<%=nom1%>" style="font-size:8pt;font-family:arial,verdana" disabled></td>
<td width="25%" align="right">
<font face="tahoma" size="1" color="<%=session.contents("color_text")%>">Segundo Nombre</font></td>
<td width="25%" align="left">
<input type="text" id="txtnombre2" name="txtnombre2" maxlength="30" size="30" onblur="ConvierteMay(this)" value="<%=nom2%>" style="font-size:8pt;font-family:arial,verdana" disabled></td>
</tr>
<tr>
<td width="25%" align="left">
<font face="tahoma" size="1" align="left" color="<%=session.contents("color_text")%>">E-mail</font></td>
<td width="75%" colspan=3 align="left">
<input type="text" id="txtemail" name="txtemail" maxlength="60" size="60" onblur="chkEmail(this)" value="<%=email%>" style="font-size:8pt;font-family:arial,verdana" disabled></td>
</tr>
<%
'Query que obtiene los roles del usuario.
Dim sqlfil
Dim rs2
Dim rsig1
set rs2= Server.CreateObject ("consulta_gral_" & Application("uv") & ".Reportes_" & Application("uv"))
sqlfil = ""
sqlfil = sqlfil & " select id_rol, descripcion "
sqlfil = sqlfil & " from trc_roles  "
If rs2.reportes(sqlfil,rsig1) Then    %>
</tr>
<td width="25%" align="left">
<font face="tahoma" size="1" color="<%=session.contents("color_text")%>">Seleccione uno de los roles para modificar al usuario:</font>
<td width="25%" align="left"><select id="selRol" name="selRol" onchange="verif_rol()">
<%
if cint(countRol) > 0 then %>
<option value="0">-- Roles --</option>
```

```
<%do while not rsRoles.EOF%>
<option value="<%=rsRoles(0)%>"><%=rsRoles(1)%></option>
<%rsRoles.movenext
loop
else  %>
<option value="0">-- No tiene asignado ningún Rol --</option>
<%end if  %>
</select>
</td>
<td width="25%" align="left"></td>
<td width="25%" align="left"></td>
</tr>
</table>
<br /><br />
<center>
<input type="button" value="Modificar Usuario" id="btnaceptar" name="btnaceptar" title="Modificar el Usuario con el Rol seleccionado" onclick="return btnaceptar_onclick()" <%if cint(countRol) = 0 then %>disabled<%end if%>>   
<input type="button" value="Cancelar" id="botcancel" name="botcancel" title="Cancelar" onclick="abc()">
</center>
<%end if  %>
</center>
</form>
```